aws

Guide du développeur V2

Amazon Lex

Amazon Lex: Guide du développeur V2

Table of Contents

Qu'est-ce qu'Amazon Lex V2 ?

Amazon Lex V2 est un service AWS permettant de créer des interfaces conversationnelles pour des applications utilisant la voix et le texte. Amazon Lex V2 fournit les fonctionnalités avancées et la flexibilité de la compréhension du langage naturel (NLU) et de la reconnaissance vocale automatique (ASR) afin que vous puissiez créer des expériences utilisateur très attrayantes avec des interactions réalistes et conversationnelles, et créer de nouvelles catégories de produits.

Amazon Lex V2 permet à tout développeur de créer rapidement des robots conversationnels. Avec Amazon Lex V2, aucune expertise en deep learning n'est requise : pour créer un bot, vous devez spécifier le flux de conversation de base dans la console Amazon Lex V2. Amazon Lex V2 gère la boîte de dialogue et ajuste de manière dynamique les réponses de la conversation. A l'aide de la console, vous pouvez créer, tester et publier votre texte ou faire parler le chatbot. Ensuite, vous pouvez ajouter les interfaces de conversation des bots sur les appareils mobiles, applications web et plateformes de conversation (par exemple, Facebook Messenger).

Amazon Lex V2 permet l'intégration à de nombreux autres services de la plateforme AWS, notamment Amazon Connect, Amazon Comprehend et Amazon Kendra. AWS Lambda L'intégration à Lambda permet aux robots d'accéder à des connecteurs d'entreprise sans serveur prédéfinis pour établir des liens avec des données dans des applications SaaS telles que Salesforce.

Pour les robots créés après le 17 août 2022, vous pouvez utiliser le branchement conditionnel pour contrôler le flux de conversation avec votre bot. Avec le branchement conditionnel, vous pouvez créer des conversations complexes sans avoir à écrire de code Lambda.

Amazon Lex V2 offre les avantages suivants :

- Simplicité : Amazon Lex V2 vous guide dans l'utilisation de la console pour créer votre propre bot en quelques minutes. Vous fournissez quelques exemples de phrases et Amazon Lex V2 crée un modèle complet de langage naturel grâce auquel le bot peut interagir en utilisant la voix et le texte pour poser des questions, obtenir des réponses et effectuer des tâches sophistiquées.

- Technologies d'apprentissage profond démocratisées : Amazon Lex V2 fournit des technologies ASR et NLU pour créer un système de compréhension du langage vocal (SLU). Grâce à SLU, Amazon Lex V2 prend en charge la saisie vocale et textuelle en langage naturel, comprend l'intention qui sous-tend la saisie et répond à l'intention de l'utilisateur en invoquant la fonction métier appropriée.

La reconnaissance vocale et la compréhension du langage naturel font partie des problèmes les plus difficiles à résoudre en informatique, nécessitant l'entraînement d'algorithmes d'apprentissage profond sophistiqués sur d'énormes quantités de données et d'infrastructures. Amazon Lex V2 met les technologies de deep learning à la portée de tous les développeurs. Les robots Amazon Lex V2 convertissent la parole entrante en texte et comprennent l'intention de l'utilisateur afin de générer une réponse intelligente. Vous pouvez ainsi vous concentrer sur la création de robots apportant une valeur ajoutée à vos clients et définir de toutes nouvelles catégories de produits grâce à des interfaces conversationnelles.

- Déploiement et mise à l'échelle simplifiés : avec Amazon Lex V2, vous pouvez créer, tester et déployer vos robots directement depuis la console Amazon Lex V2. Amazon Lex V2 vous permet de publier vos robots vocaux ou textuels afin de les utiliser sur des appareils mobiles, des applications Web et des services de chat (par exemple, Facebook Messenger). Amazon Lex V2 évolue automatiquement. Vous n'avez pas à vous soucier du provisionnement du matériel et de la gestion de l'infrastructure pour optimiser votre expérience de bot.

- Intégration intégrée à la plateforme AWS : Amazon Lex V2 fonctionne de manière native avec d'autres services AWS, tels qu'AWS LambdaAmazonCloudWatch. Vous pouvez profiter de la puissance de la plateforme AWS pour la sécurité, la surveillance, l'authentification utilisateur, la logique métier, le stockage et le développement d'applications mobiles.

- Rentabilité : avec Amazon Lex V2, il n'y a pas de coûts initiaux ni de frais minimaux. Vous êtes facturé uniquement pour les demandes textuelles ou vocales qui sont effectuées. La pay-as-you-go tarification et le faible coût par demande font du service un moyen rentable de créer des interfaces conversationnelles. Avec le niveau gratuit d'Amazon Lex V2, vous pouvez facilement essayer Amazon Lex V2 sans aucun investissement initial.

Payer pour Amazon Lex

Amazon Lex V2 ne vous facture que les requêtes de texte ou de reconnaissance vocale que vous effectuez. Ce modèle vous offre un service à coût variable qui peut évoluer avec votre activité tout en

vous offrant les avantages financiers de l'infrastructure AWS. Pour plus d'informations, consultez la page de tarification d'Amazon Lex.

Lorsque vous vous inscrivez àAWS, votre AWS compte est automatiquement inscrit à tous les servicesAWS, y compris Amazon Lex. Toutefois, seuls les services que vous utilisez vous sont facturés. Si vous êtes un nouveau client Amazon Lex, vous pouvez commencer à utiliser Amazon Lex gratuitement. Pour plus d'informations, consultez l'offre gratuite d'AWS.

Pour consulter votre facture, dirigez-vous vers le Tableau de bord de gestion des coûts et de la facturation dans la AWS Billing and Cost Managementconsole. Pour en savoir plus sur la facturation de Compte AWS, consultez le guide de l'utilisateur AWS Billing. Pour toute question relative à la facturation AWS et aux Comptes AWS, contactez le support AWS.

Utilisez-vous Amazon Lex V2 pour la première fois ?

Si vous utilisez Amazon Lex V2 pour la première fois, nous vous recommandons de lire les sections suivantes dans l'ordre :

1. Comment ça marche— Cette section présente Amazon Lex V2 et les fonctionnalités que vous utilisez pour créer un chatbot.
2. Démarrez avec Amazon Lex V2— Dans cette section, vous allez configurer votre compte et tester Amazon Lex V2.
3. Référence d'API : cette section contient des détails sur les opérations de l'API.

Fonctionnalités les plus récentes

Découvrez les dernières fonctionnalités d'Amazon Lex V2 ci-dessous :

Rubriques

Support régional pour AWS GovCloud (US-Ouest)

Amazon Lex V2 est désormais disponible en AWS GovCloud (ouest des États-Unis).

- Points de terminaison et quotas Amazon Lex

Fonctionnalités d'IA générative pour Amazon Lex V2

Amazon Lex V2 vous permet désormais de tirer parti des capacités d'intelligence artificielle générative d'Amazon Bedrock pour votre bot.

- Générateur de bots descriptif
  - Quel est le nouveau post
  - Documentation
- Résolution des créneaux assistée
  - Quel est le nouveau post
  - Documentation
- Génération d'énoncés
  - Quel est le nouveau post
  - Documentation
- `AMAZON.QnAIntent`(FAQ conversationnelle)
  - Quel est le nouveau post
  - Documentation
- AWS Article de blog sur le Machine Learning

Emplacement intégré Amazon.Confirmation pour la désambiguïsation Oui/Non/Peut-être/Je ne sais pas.

Amazon Lex V2 propose désormais un emplacement `AMAZON.Confimation` intégré pour améliorer la précision de la confirmation du créneau et des réponses Oui/Non/Peut-être/Je ne sais pas.

- Documentation

Mesurer les performances de l'entreprise avec Analytics

Amazon Lex V2 offre désormais aux utilisateurs la possibilité de consulter les performances des intentions et des emplacements sur le tableau de bord Analytics.

- Quel est le nouveau post
- Documentation

Évaluation des performances des robots avec Test Workbench

Amazon Lex V2 offre désormais aux utilisateurs la possibilité de créer et d'exécuter des ensembles de tests pour mesurer les performances des robots et améliorer les indicateurs des robots.

- Quel est le nouveau post
- Documentation
- AWS Article de blog sur le Machine Learning

Modèles de robots spécifiques à la verticale

Amazon Lex V2 propose désormais aux utilisateurs des modèles de bots prédéfinis avec des flux de ready-to-use conversation ainsi que des données de formation et des instructions de dialogue, pour les modalités vocales et de chat.

- Quel est le nouveau post
- Documentation

Réseau de bots

Amazon Lex V2 offre désormais aux utilisateurs la possibilité de combiner plusieurs robots au sein d'un même réseau et d'acheminer les demandes vers le bot approprié en fonction des informations saisies par les utilisateurs.

- Quel est le nouveau post
- Documentation

Générateur de conversation visuel

Amazon Lex V2 propose désormais un générateur de conversation par glisser-déposer permettant de concevoir et de visualiser facilement des parcours de conversation en utilisant les intentions dans un environnement visuel riche.

- Quel est le nouveau post
- Documentation
- AWS Article de blog sur le Machine Learning

Type de fente composite

Amazon Lex V2 offre désormais aux utilisateurs la possibilité de combiner plusieurs emplacements dans un emplacement composite à l'aide d'expressions logiques.

- Quel est le nouveau post
- Documentation

Branchement conditionnel

Amazon Lex V2 offre désormais aux utilisateurs la possibilité de rédiger des conditions afin de mieux contrôler le chemin emprunté par les clients lors d'une conversation avec votre bot.

- Quel est le nouveau post
- Documentation

Concepteur de chatbot automatisé

Amazon Lex V2 offre désormais aux utilisateurs la possibilité de concevoir automatiquement un chatbot à partir des transcriptions de conversation. Lisez les exemples d'utilisation.

- Quel est le nouveau post
- Documentation
- AWS Article de blog sur le Machine Learning
- Page du concepteur de Chatbot automatisé Amazon Lex

Conseils d'exécution

Amazon Lex V2 offre désormais aux utilisateurs la possibilité de configurer des indices d'exécution pour améliorer la reconnaissance des phrases et ainsi améliorer l'obtention des valeurs des créneaux.

- Quel est le nouveau post
- Documentation

Vocabulaire personnalisé

Amazon Lex V2 offre désormais aux utilisateurs la possibilité de créer un vocabulaire personnalisé, une liste de phrases pouvant inclure des noms propres ou des mots spécifiques à un domaine, afin qu'Amazon Lex V2 puisse les reconnaître dans l'entrée audio.

- Quel est le nouveau post
- Documentation
- AWS Article de blog sur le Machine Learning

Type de slot grammatical

Amazon Lex V2 offre désormais aux utilisateurs la possibilité de créer des grammaires au format XML conformément à la spécification grammaticale de reconnaissance vocale (SRGS) afin de collecter des informations lors d'une conversation.

- Quel est le nouveau post
- Documentation
- Article de blog AWS Machine Learning

Comment ça marche

Amazon Lex V2 vous permet de créer des applications à l'aide d'une interface texte ou vocale pour une conversation avec un utilisateur. Voici les étapes habituelles d'utilisation d'Amazon Lex V2 :

1. Créez un bot et ajoutez une ou plusieurs langues. Configurez le bot de manière à ce qu'il comprenne l'objectif de l'utilisateur, qu'il engage une conversation avec l'utilisateur pour obtenir des informations et qu'il réponde à l'intention de l'utilisateur.
2. Testez le bot. Vous pouvez utiliser le client de fenêtre de test fourni par la console Amazon Lex V2.
3. Publiez une version et créez un alias.
4. Déployez le bot. Vous pouvez déployer le bot sur vos propres applications ou plateformes de messagerie telles que Facebook Messenger ou Slack

Avant de commencer, familiarisez-vous avec les concepts de base et la terminologie suivants d'Amazon Lex V2 :

- Bot — Un bot exécute des tâches automatisées telles que la commande d'une pizza, la réservation d'un hôtel, la commande de fleurs, etc. Un bot Amazon Lex V2 est alimenté par des fonctionnalités de reconnaissance vocale automatique (ASR) et de compréhension du langage naturel (NLU).

  Les robots Amazon Lex V2 peuvent comprendre les entrées utilisateur sous forme de texte ou de parole et converser en langage naturel.
- Langue : un bot Amazon Lex V2 peut converser dans une ou plusieurs langues. Chaque langue étant indépendante des autres, vous pouvez configurer Amazon Lex V2 pour converser avec un utilisateur en utilisant des mots et des phrases natifs. Pour plus d'informations, veuillez consulter Langues et paramètres régionaux pris en charge par Amazon Lex V2.
- Intention : une intention représente une action que l'utilisateur souhaite effectuer. Vous créez un bot pour prendre en charge une ou plusieurs intentions connexes. Par exemple, vous pouvez créer une intention qui commande des pizzas et des boissons. Pour chaque intention, vous fournissez les informations obligatoires suivantes :
  - Nom de l'intention : nom descriptif de l'intention. Par exemple, **`OrderPizza`**.
  - Exemples d'énoncés : comment un utilisateur peut transmettre son intention. Par exemple, un utilisateur peut dire « Puis-je commander une pizza » ou « Je veux commander une pizza ».

- Comment atteindre l'intention : comment vous souhaitez réaliser l'intention une fois que l'utilisateur a fourni les informations nécessaires. Nous vous recommandons de créer une fonction Lambda pour atteindre cet objectif.

  Vous pouvez éventuellement configurer l'intention pour qu'Amazon Lex V2 renvoie les informations à l'application cliente pour l'expédition nécessaire.

Outre les intentions personnalisées, Amazon Lex V2 fournit des intentions intégrées pour configurer rapidement votre bot. Pour plus d'informations, veuillez consulter Intentions prédéfinies.

Amazon Lex inclut toujours une intention de repli pour chaque bot. L'intention de repli est utilisée chaque fois qu'Amazon Lex ne parvient pas à déduire l'intention de l'utilisateur. Pour plus d'informations, veuillez consulter AMAZON.FallbackIntent.

- Emplacement : une intention peut ne nécessiter aucun emplacement ou plusieurs paramètres. Vous ajoutez des options dans le cadre de la configuration d'intention. Lors de l'exécution, Amazon Lex V2 invite l'utilisateur à saisir des valeurs d'emplacement spécifiques. L'utilisateur doit fournir des valeurs pour tous les emplacements requis avant qu'Amazon Lex V2 ne puisse atteindre son objectif.

  Par exemple, l'`OrderPizza`intention exige des emplacements tels que la taille, le type de croûte et le nombre de pizzas. Pour chaque emplacement, vous indiquez le type d'emplacement et une ou plusieurs instructions qu'Amazon Lex V2 envoie au client pour obtenir des valeurs de la part de l'utilisateur. Un utilisateur peut répondre en utilisant une valeur d'emplacement contenant des mots supplémentaires, tels que « une grosse pizza, s'il vous plaît » ou « restons-en à une petite ». Amazon Lex V2 comprend toujours la valeur de l'emplacement.

- Type d'emplacement — Chaque emplacement possède un type. Vous pouvez créer votre propre type d'emplacement ou utiliser des types d'emplacement intégrés. Par exemple, vous pouvez créer et utiliser les types d'options suivants pour l'intention `OrderPizza` :

  - Taille – avec des valeurs d'énumération `Small`, `Medium` et `Large`.
  - Pâte – avec des valeurs d'énumération `Thick` et `Thin`.

  Amazon Lex V2 propose également des types d'emplacements intégrés. Par exemple, `AMAZON.Number` est un type d'option prédéfini que vous pouvez utiliser avec le nombre de pizzas commandées. Pour plus d'informations, veuillez consulter Intentions prédéfinies.

- Version : une version est un instantané numéroté de votre travail que vous pouvez publier pour l'utiliser dans différentes parties de votre flux de travail, telles que le développement, le déploiement bêta et la production. Une fois que vous avez créé une version, vous pouvez utiliser

un bot tel qu'il existait au moment de la création de la version. Une fois que vous avez créé une version, elle reste inchangée pendant que vous continuez à travailler sur votre application.

- Alias — Un alias est un pointeur vers une version spécifique d'un bot. Un alias vous permet de mettre à jour la version utilisée par vos applications clientes. Par exemple, vous pouvez faire pointer un alias vers la version 1 de votre bot. Lorsque vous êtes prêt à mettre à jour le bot, vous publiez la version 2 et vous modifiez l'alias pour qu'il pointe vers la nouvelle version. Comme vos applications utilisent l'alias au lieu d'une version spécifique, tous vos clients obtiennent les nouvelles fonctionnalités sans avoir besoin d'être mis à jour.

Pour obtenir la liste des AWS régions dans lesquelles Amazon Lex V2 est disponible, consultez la section Points de terminaison et quotas Amazon Lex V2 dans la référence générale d'Amazon Web Services.

Langues et paramètres régionaux pris en charge par Amazon Lex V2

Amazon Lex V2 prend en charge un grand nombre de langues et de paramètres régionaux. Les langues prises en charge, les fonctionnalités qui prennent en charge ces langues et les conseils spécifiques à chaque langue pour améliorer les performances de votre bot sont fournis dans cette rubrique.

Langues et paramètres régionaux pris en charge

Amazon Lex V2 prend en charge les langues et les paramètres régionaux suivants.

Code	Langue et paramètres régionaux
ar_AE	Golfe arabe (Émirats arabes unis)
CA_fr	Catalan (Espagne)
fr_AT	Allemand (Autriche)
fr_FR	Allemand (Allemagne)
fr_AU	Anglais (Australie)
fr_GB	Anglais (Royaume-Uni)

Code	Langue et paramètres régionaux
fr_IN	Anglais (Inde)
fr_FR	Anglais (États-Unis)
fr_ZA	Anglais (Afrique du Sud)
fr_419	Espagnol (Amérique latine)
fr_FR	Espagnol (Espagne)
fr_US	Espagnol (États-Unis)
fr_FR	Finnois (Finlande)
fr_CA	Français (Canada)
fr_FR	Français (France)
Salut	Hindi (Inde)
fr_FR	Italien (Italie)
ja_JP	Japonais (Japon)
fr_FR	Coréen (Corée)
fr_NL	Néerlandais (Pays-Bas)
Non_non	Norvégien (Norvège)
fr_FR	Polonais (Pologne)
fr_BR	Portugais (Brésil)
fr_FR	Portugais (Portugal)
fr_SE	Suédois (Suède)
zh_FR	Mandarin (RPC)

Code	Langue et paramètres régionaux
zh_HK	Cantonais (Hong Kong)

Langues et paramètres régionaux pris en charge par les fonctionnalités d'Amazon Lex V2

Le tableau suivant répertorie les fonctionnalités d'Amazon Lex V2 qui sont limitées à certaines langues et à certains paramètres régionaux. Toutes les autres fonctionnalités d'Amazon Lex V2 sont prises en charge dans toutes les langues et tous les paramètres régionaux.

Fonction	Langues et paramètres régionaux pris en charge
AMAZON. AlphaNumeric	Toutes les langues et tous les paramètres régionaux sauf le coréen (ko_KR)
AMAZON.KendraSearchIntent	Anglais (États-Unis) (en_US)
Améliorer la reconnaissance vocale grâce à un vocabulaire personnalisé	Anglais (UK) (en_GB) Anglais (États-Unis) (en_US)
Concepteur de chatbot automatisé	Anglais (États-Unis) (en_US)
Disponibilité dans les Régions	Les langues et paramètres régionaux suivants ne sont pas disponibles dans les régions Asie-Pacifique (Singapour) (ap-southeast-1) et Afrique (Le Cap) (ap-south-1) : • Golfe arabe (Émirats arabes unis) (ar_AE) • Catalan (Espagne) (ca_ES) • Finnois (Finlande) (fi_FI) • Hindi (Inde) (hi_in) • Néerlandais (Pays-Bas) (NL_NL) • Norvégien (Norvège) (No_no)

Fonction	Langues et paramètres régionaux pris en charge
	• Polonais (pl_PL) • Portugais (Brésil) (pt_BR) • Portugais (Portugal) (pt_PT) • Suédois (sv_SE) • Mandarine (PRC) (zh_CN) • Cantonais (Hong Kong) (zh_HK)
Définition du contexte de l'intention	Anglais (États-Unis) (en_US)
Type d'emplacement de grammaire	Anglais (Australie) (en_AU) Anglais (UK) (en_GB) Anglais (États-Unis) (en_US)
Utilisation de plusieurs valeurs dans un emplacement	Anglais (États-Unis) (en_US)
Amélioration de la reconnaissance des valeurs des créneaux grâce à des indices d'exécution	Anglais (UK) (en_GB) Anglais (États-Unis) (en_US)
Capture de valeurs de créneaux avec des styles d'orthographe	Anglais (Australie) (en_AU) Anglais (UK) (en_GB) Anglais (États-Unis) (en_US)
Utilisation des scores de confiance	Anglais (UK) (en_GB) Anglais (États-Unis) (en_US)

Conseils linguistiques pour Amazon Lex V2

Pour améliorer les performances de votre bot, vous devez respecter ces directives pour les langues suivantes.

Arabe

La variété d'arabe utilisée pour la formation d'Amazon Lex V2 est l'arabe du Golfe. Gardez cela à l'esprit lorsque vous fournissez des exemples d'énoncés à votre bot. Notez que l'écriture arabe s'écrit de droite à gauche.

Hindi

Amazon Lex V2 est capable de répondre aux besoins des utilisateurs finaux en hindi qui passent librement de l'hindi à l'anglais. Si vous envisagez de créer un bot prenant en charge ce changement de langue, nous vous recommandons de suivre les meilleures pratiques suivantes :

- Dans la définition du bot, écrivez les mots anglais en caractères latins.
- Au moins 50 % de vos exemples d'énoncés doivent représenter un changement de langue au sein d'une même phrase. Dans ces énoncés, utilisez l'écriture devanagari pour les mots hindi et l'écriture latine pour les mots anglais (par exemple, « carnet de billets »). «).
- Si vous vous attendez à ce que les utilisateurs communiquent avec le bot en utilisant des mots hindi en script latin ou des mots anglais en script devanagari, vous devez inclure des exemples de mots hindi en script latin (par exemple, « mujhe ek ticket book karni hai ») et de mots anglais en script devanagari (par exemple, « ») dans vos exemples d'énoncés.
- Si vous vous attendez à ce que les utilisateurs communiquent avec le bot en utilisant des phrases entièrement en hindi ou entièrement en anglais, vous devez inclure des exemples d'énoncés entièrement rédigés dans une langue (par exemple, « Je veux réserver un billet »).

Régions

Pour obtenir la liste des AWS régions dans lesquelles Amazon Lex V2 est disponible, consultez les régions et points de terminaison AWS dans leRéférences générales AWS.

Démarrez avec Amazon Lex V2

Amazon Lex V2 fournit des opérations d'API que vous pouvez intégrer à vos applications existantes. Pour obtenir la liste des opérations prises en charge, consultez la référence de l'API. Vous pouvez utiliser les options suivantes :

- SDK AWS : lorsque vous utilisez les kits SDK, vos demandes adressées à Amazon Lex V2 sont automatiquement signées et authentifiées à l'aide des informations d'identification que vous fournissez. Nous vous recommandons d'utiliser un SDK pour créer votre application.
- AWS CLI— Vous pouvez utiliser le AWS CLI pour accéder à n'importe quelle fonctionnalité d'Amazon Lex V2 sans avoir à écrire de code.
- Console AWS : la console est le moyen le plus simple de commencer à tester et à utiliser Amazon Lex V2

Si vous utilisez Amazon Lex V2 pour la première fois, nous vous recommandons de lire Comment ça marche d'abord.

Rubriques

Étape 1 : Configuration d'un AWS compte et création d'un utilisateur administrateur

Avant d'utiliser Amazon Lex V2 pour la première fois, effectuez les tâches suivantes :

1. S'inscrire à AWS
2. Créer un utilisateur IAM

S'inscrire à AWS

Si vous avez déjà un compte AWS, ignorez cette étape.

Lorsque vous vous inscrivez à Amazon Web Services (AWS), votre AWS compte est automatiquement inscrit à tous les servicesAWS, y compris Amazon Lex V2. Seuls les services que vous utilisez vous sont facturés.

Avec Amazon Lex V2, vous ne payez que pour les ressources que vous utilisez. Si vous êtes un nouveau AWS client, vous pouvez commencer à utiliser Amazon Lex V2 gratuitement. Pour plus d'informations, consultez la page Niveau d'offre gratuite d'AWS.

Si vous possédez déjà un compte AWS, passez à la prochaine étape. Si vous n'avez pas de compte AWS, observez la procédure suivante pour en créer un.

Créer un compte AWS

1. Ouvrez https://portal.aws.amazon.com/billing/signup.
2. Suivez les instructions en ligne.

   Dans le cadre de la procédure d'inscription, vous recevrez un appel téléphonique et vous saisirez un code de vérification en utilisant le clavier numérique du téléphone.

   Lorsque vous souscrivez à un Compte AWS, un Utilisateur racine d'un compte AWS est créé. Par défaut, seul l'utilisateur root a accès à l'ensemble des Services AWS et des ressources de ce compte. En tant que bonne pratique de sécurité, attribuer un accès administratif à un utilisateur administratif, et utilisez uniquement l'utilisateur root pour effectuer tâches nécessitant un accès utilisateur root.

Notez l'ID de votre compte AWS, car vous en aurez besoin lors de la prochaine tâche.

Créer un utilisateur IAM

Les servicesAWS, tels qu'Amazon Lex V2, nécessitent que vous fournissiez des informations d'identification lorsque vous y accédez afin que le service puisse déterminer si vous êtes autorisé à accéder aux ressources qui lui appartiennent.

Créez un compte utilisateur IAM pour accéder à votre compte Amazon Lex V2 :

- Utilisez AWS Identity and Access Management (IAM) pour créer un utilisateur IAM
- Ajouter l'utilisateur à un groupe IAM avec des autorisations administratives
- Accordez des autorisations administratives à l'utilisateur IAM que vous avez créé.

Vous pouvez ensuite y accéder à AWS l'aide d'une URL spéciale et des informations d'identification de l'utilisateur IAM.

Les exercices de mise en route de ce guide présument que l'utilisateur (`adminuser`) dispose de privilèges d'administrateur. Suivez la procédure pour créer `adminuser` dans votre compte.

Pour créer un administrateur et vous connecter à la console

1. Créez un administrateur appelé `adminuser` dans votre compte AWS. Pour obtenir des instructions, consultez la section Création de votre premier utilisateur IAM et de votre premier groupe d'administrateurs dans le Guide de l'utilisateur IAM.
2. En tant qu'utilisateur, vous pouvez vous connecter à AWS Management Console à l'aide d'une URL spéciale. Pour plus d'informations, consultez la section Comment les utilisateurs se connectent à votre compte dans le Guide de l'utilisateur IAM.

Pour plus d'informations sur IAM, consultez les ressources suivantes :

- AWS Identity and Access Management (IAM)
- Prise en main
- Guide de l'utilisateur IAM

Accorder un accès par programmation

Les utilisateurs ont besoin d'un accès programmatique s'ils souhaitent interagir avec AWS en dehors de la AWS Management Console. La manière d'octroyer un accès par programmation dépend du type d'utilisateur qui accède à AWS.

Pour accorder aux utilisateurs un accès programmatique, choisissez l'une des options suivantes.

Quel utilisateur a besoin d'un accès programmatique ?	Pour	Bit
Identité de la main-d'œuvre (Utilisateurs gérés dans IAM Identity Center)	Utilisez des informations d'identification temporaires pour signer des demandes par programmation destinées à	Suivez les instructions de l'interface que vous souhaitez utiliser. • Pour l'AWS CLI, veuillez consulter la rubrique

Quel utilisateur a besoin d'un accès programmatique ?	Pour	Bit
	l'AWS CLI, aux kits SDK AWS ou aux API AWS.	Configuration de l'AWS CLI pour l'utilisation d'AWS IAM Identity Center dans le Guide de l'utilisateur AWS Command Line Interface. • Pour les kits SDK et les outils AWS ainsi que les API AWS, veuillez consulter la rubrique Authentification IAM Identity Center dans le Guide de référence des kits SDK et des outils AWS.
IAM	Utilisez des informations d'identification temporaires pour signer des demandes par programmation destinées à l'AWS CLI, aux kits SDK AWS ou aux API AWS.	Suivez les instructions de la section Utilisation d'informations d'identification temporaires avec des ressources AWS dans le Guide de l'utilisateur IAM.

Quel utilisateur a besoin d'un accès programmatique ?	Pour	Bit
IAM	(Non recommandé) Utilisez des informations d'identification à long terme pour signer des demandes par programmation destinées à l'AWS CLI, aux kits SDK AWS ou aux API AWS.	Suivez les instructions de l'interface que vous souhaitez utiliser. • Pour l'AWS CLI, veuillez consulter la rubrique Authentification à l'aide des informations d'identification d'utilisateur IAM dans le Guide de l'utilisateur AWS Command Line Interface. • Pour les kits SDK et les outils AWS, veuillez consulter la rubrique Authentification à l'aide d'informations d'identification à long terme dans le Guide de référence des kits SDK et des outils AWS. • Pour les API AWS, veuillez consulter la rubrique Gestion des clés d'accès pour les utilisateurs IAM dans le Guide de l'utilisateur IAM.

Étape suivante

Étape 2 : Démarrage (console)

Étape 2 : Démarrage (console)

La manière la plus simple d'apprendre à utiliser Amazon Lex V2 est d'utiliser la console. Pour vous aider à faire vos premiers pas, nous avons créé les exercices suivants, qui utilisent tous la console :

- Exercice 1 — Créez un bot Amazon Lex V2 à l'aide d'un blueprint, un bot prédéfini qui fournit toute la configuration de bot nécessaire. Vous n'effectuez qu'un travail minimum pour tester la configuration de bout en bout.
- Exercice 2 — Passez en revue les structures JSON envoyées entre votre application cliente et un bot Amazon Lex V2.

Rubriques

Exercice 1 : créer un bot à partir d'un exemple

Dans cet exercice, vous allez créer votre premier bot Amazon Lex V2 et le tester dans la console Amazon Lex V2. Pour cet exercice, vous utiliserez cet OrderFlowersexemple.

Exemple d'aperçu

Vous utilisez cet OrderFlowersexemple pour créer un bot Amazon Lex V2. Pour plus d'informations sur la structure d'un bot, consultezComment ça marche.

- Intention — OrderFlowers
- Types d'option – un type d'option personnalisé appelé `FlowerTypes` avec des valeurs d'énumération : `roses`, `lilies` et `tulips`.
- Options – L'intention nécessite les informations suivantes (c'est à dire des options) pour que le bot puisse traiter l'intention.
  - `PickupTime` (type prédéfini AMAZON.TIME)
  - `FlowerType`(type FlowerTypes personnalisé)
  - `PickupDate` (type prédéfini AMAZON.DATE)
- Enoncé – Les exemples d'énoncés suivants indiquent l'intention de l'utilisateur :
  - « I would like to pick up flowers. »

  - « "I would like to order some flowers. »
- Invites – une fois que le bot a identifié l'intention, il utilise les invites suivantes pour indiquer les options :
  - Invite de l'option `FlowerType` – « Quel type de fleurs souhaitez-vous commander ? »
  - Demande de `PickupDate` créneau — « Quel jour voulez-vous que le {FlowerType} soit retiré ? »
  - Demander le `PickupTime` créneau — « À quelle heure voulez-vous que le {FlowerType} soit retiré ? »
  - Déclaration de confirmation — « OK, votre {FlowerType} sera prêt à être retiré le {PickupTime} le {PickupDate}. Cela vous convient-il ? »

Pour créer un bot Amazon Lex V2 (console)

1. Connectez-vous à la console Amazon Lex AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/lex/.
2. Choisissez Create bot.
3. Pour la méthode de création, choisissez Commencer par un exemple.
4. Dans la section Exemples de robots, OrderFlowerschoisissez dans la liste.
5. Dans la section Configuration du bot, donnez un nom au bot et une description facultative. Le nom doit être unique dans votre compte.
6. Dans la section Autorisations, choisissez Créer un nouveau rôle avec des autorisations Amazon Lex de base. Cela créera un rôle AWS Identity and Access Management (IAM) doté des autorisations dont Amazon Lex V2 a besoin pour exécuter votre bot.
7. Dans la section Loi sur la protection de la vie privée en ligne des enfants (COPPA), faites le choix approprié.
8. Dans les sections Expiration de session et Paramètres avancés, conservez les valeurs par défaut.
9. Choisissez Suivant. Amazon Lex V2 crée votre bot.

Après avoir créé votre bot, vous devez ajouter une ou plusieurs langues prises en charge par le bot. Une langue contient les intentions, les types d'emplacements et les emplacements que le bot utilise pour converser avec les utilisateurs.

Pour ajouter une langue à un bot

1. Dans la section Langue, choisissez une langue prise en charge et ajoutez une description.
2. Conservez les champs de seuil de confiance relatifs à l'interaction vocale et à la classification des intentions avec leurs valeurs par défaut.
3. Choisissez OK pour ajouter la langue au bot.

Après avoir sélectionné OK, la console ouvre l'éditeur d'intention. Vous pouvez utiliser l'éditeur d'intention pour examiner les intentions utilisées par le bot. Lorsque vous avez terminé d'examiner le bot, vous pouvez le tester.

Pour tester le OrderFlowers bot

1. Choisissez Créer en haut de la page. Attendez que le bot soit créé.
2. Lorsque la génération est terminée, choisissez Test pour ouvrir la fenêtre de test.
3. Testez le bot. Commencez la conversation par l'un des exemples d'énoncés, comme « J'aimerais cueillir des fleurs ».

Étapes suivantes

Maintenant que vous avez créé votre premier bot à l'aide d'un modèle, vous pouvez utiliser la console pour créer votre propre bot. Pour obtenir des instructions sur la création d'un bot personnalisé et pour plus d'informations sur la création de robots, consultezConstruire des robots.

Exercice 2 : Passez en revue le flux de conversation

Dans cet exercice, vous allez passer en revue les structures JSON qui sont envoyées entre votre application cliente et le bot Amazon Lex V2 que vous avez créé[Exercice 1 : créer un bot à partir d'un exemple](). La conversation utilise l'[RecognizeText]()opération pour générer les structures JSON. [RecognizeUtterance]()Renvoie les mêmes informations que les en-têtes HTTP de la réponse.

Les structures JSON sont divisées à chaque tour de conversation. Un tour est une demande de l'application cliente et une réponse du bot.

Tourner 1

Au premier tour de la conversation, l'application cliente lance la conversation avec votre bot. L'URI et le corps de la requête fournissent des informations sur la requête.

```
POST /bots/botId/botAliases/botAliasId/botLocales/localeId/sessions/sessionId/text
 HTTP/1.1
Content-type: application/json

{
    "text": "I would like to order flowers"
}
```

- L'URI identifie le bot avec lequel l'application cliente communique. Il comprend également un identifiant de session généré par l'application cliente qui identifie une conversation spécifique entre un utilisateur et le robot.
- Le corps de la demande contient le texte que l'utilisateur a saisi dans l'application cliente. Dans ce cas, seul le texte est envoyé, mais votre application peut envoyer des informations supplémentaires, telles que les attributs de la demande ou l'état de la session. Pour plus d'informations, consultez l'RecognizeTextopération.

À partir de là`text`, Amazon Lex V2 détecte l'intention de l'utilisateur de commander des fleurs. Amazon Lex V2 choisit l'un des emplacements de l'intention (`FlowerType`) et l'une des invites correspondant à cet emplacement, puis envoie la réponse suivante à l'application cliente. Le client affiche la réponse à l'utilisateur.

```
{
    "interpretations": [
        {
            "intent": {
                "confirmationState": "None",
                "name": "OrderFlowers",
                "slots": {
                    "FlowerType": null,
                    "PickupDate": null,
                    "PickupTime": null
                },
                "state": "InProgress"
            },
            "nluConfidence": {
                "score": 0.95
            }
        },
        {
```

```
            "intent": {
                "name": "FallbackIntent",
                "slots": {}
            }
        }
    ],
    "messages": [
        {
            "content": "What type of flowers would you like to order?",
            "contentType": "PlainText"
        }
    ],
    "sessionId": "bf445a49-7165-4fcd-9a9c-a782493fba5c",
    "sessionState": {
        "dialogAction": {
            "slotToElicit": "FlowerType",
            "type": "ElicitSlot"
        },
        "intent": {
            "confirmationState": "None",
            "name": "OrderFlowers",
            "slots": {
                "FlowerType": null,
                "PickupDate": null,
                "PickupTime": null
            },
            "state": "InProgress"
        },
        "originatingRequestId": "9e8add70-4106-4a10-93f5-2ce2cb959e5f"
    }
}
```

Tourner 2

Au tour 2, l'utilisateur répond à l'invite du bot Amazon Lex V2 au tour 1 avec une valeur qui remplit la`FlowerType` case.

```
{
    "text": "1 dozen roses"
}
```

La réponse pour le tour 2 indique que le`FlowerType` créneau est rempli et invite à obtenir la valeur de l'emplacement suivant.


```
{
    "interpretations": [
        {
            "intent": {
                "confirmationState": "None",
                "name": "OrderFlowers",
                "slots": {
                    "FlowerType": {
                        "value": {
                            "interpretedValue": "dozen roses",
                            "originalValue": "dozen roses",
                            "resolvedValues": []
                        }
                    },
                    "PickupDate": null,
                    "PickupTime": null
                },
                "state": "InProgress"
            },
            "nluConfidence": {
                "score": 0.98
            }
        },
        {
            "intent": {
                "name": "FallbackIntent",
                "slots": {}
            }
        }
    ],
    "messages": [
        {
            "content": "What day do you want the dozen roses to be picked up?",
            "contentType": "PlainText"
        }
    ],
    "sessionId": "bf445a49-7165-4fcd-9a9c-a782493fba5c",
    "sessionState": {
        "dialogAction": {
```

```
            "slotToElicit": "PickupDate",
            "type": "ElicitSlot"
        },
        "intent": {
            "confirmationState": "None",
            "name": "OrderFlowers",
            "slots": {
                "FlowerType": {
                    "value": {
                        "interpretedValue": "dozen roses",
                        "originalValue": "dozen roses",
                        "resolvedValues": []
                    }
                },
                "PickupDate": null,
                "PickupTime": null
            },
            "state": "InProgress"
        },
        "originatingRequestId": "9e8add70-4106-4a10-93f5-2ce2cb959e5f"
    }
}
```

Tourner 3

Au tour 3, l'utilisateur répond à l'invite du bot Amazon Lex V2 au tour 2 avec une valeur qui remplit la`PickupDate` case.

```
{
    "text": "next monday"
}
```

La réponse pour le tour 3 est`PickupDate` à la`FlowerType` fois remplie et fournit une invite pour obtenir la valeur du dernier emplacement.

```
{
    "interpretations": [
        {
            "intent": {
                "confirmationState": "None",
```

```
                "name": "OrderFlowers",
                "slots": {
                    "FlowerType": {
                        "value": {
                            "interpretedValue": "dozen roses",
                            "originalValue": "dozen roses",
                            "resolvedValues": []
                        }
                    },
                    "PickupDate": {
                        "value": {
                            "interpretedValue": "2022-12-28",
                            "originalValue": "next monday",
                            "resolvedValues": [
                                "2021-01-04"
                            ]
                        }
                    },
                    "PickupTime": null
                },
                "state": "InProgress"
            },
            "nluConfidence": {
                "score": 1.0
            }
        },
        {
            "intent": {
                "name": "FallbackIntent",
                "slots": {}
            }
        }
    ],
    "messages": [
        {
            "content": "At what time do you want the 1 dozen roses to be picked up?",
            "contentType": "PlainText"
        }
    ],
    "sessionId": "bf445a49-7165-4fcd-9a9c-a782493fba5c",
    "sessionState": {
        "dialogAction": {
            "slotToElicit": "PickupTime",
            "type": "ElicitSlot"
```

```
        },
        "intent": {
            "confirmationState": "None",
            "name": "OrderFlowers",
            "slots": {
                "FlowerType": {
                    "value": {
                        "interpretedValue": "dozen roses",
                        "originalValue": "dozen roses",
                        "resolvedValues": []
                    }
                },
                "PickupDate": {
                    "value": {
                        "interpretedValue": "2021-01-04",
                        "originalValue": "next monday",
                        "resolvedValues": [
                            "2021-01-04"
                        ]
                    }
                },
                "PickupTime": null
            },
            "state": "InProgress"
        },
        "originatingRequestId": "9e8add70-4106-4a10-93f5-2ce2cb959e5f",
        "sessionAttributes": {}
    }
}
```

Virage 4

Au tour 4, l'utilisateur fournit la valeur d'emplacement finale correspondant à l'intention, à savoir l'heure à laquelle les fleurs sont cueillies.

```
{
    "text": "5 in the evening"
}
```

Dans la réponse, Amazon Lex V2 envoie une invite de confirmation à l'utilisateur pour confirmer que la commande est correcte. Le`dialogAction` est réglé sur`ConfirmIntent` et le`confirmationState` est`None`.


```
{
    "interpretations": [
        {
            "intent": {
                "confirmationState": "None",
                "name": "OrderFlowers",
                "slots": {
                    "FlowerType": {
                        "value": {
                            "interpretedValue": "dozen roses",
                            "originalValue": "dozen roses",
                            "resolvedValues": []
                        }
                    },
                    "PickupDate": {
                        "value": {
                            "interpretedValue": "2021-01-04",
                            "originalValue": "next monday",
                            "resolvedValues": [
                                "2021-01-04"
                            ]
                        }
                    },
                    "PickupTime": {
                        "value": {
                            "interpretedValue": "17:00",
                            "originalValue": "5 evening",
                            "resolvedValues": [
                                "17:00"
                            ]
                        }
                    }
                },
                "state": "InProgress"
            },
            "nluConfidence": {
                "score": 1.0
            }
```

```
        },
        {
            "intent": {
                "name": "FallbackIntent",
                "slots": {}
            }
        }
    ],
    "messages": [
        {
            "content": "Okay, your dozen roses will be ready for pickup by 17:00 on
2021-01-04.  Does this sound okay?",
            "contentType": "PlainText"
        }
    ],
    "sessionId": "bf445a49-7165-4fcd-9a9c-a782493fba5c",
    "sessionState": {
        "dialogAction": {
            "type": "ConfirmIntent"
        },
        "intent": {
            "confirmationState": "None",
            "name": "OrderFlowers",
            "slots": {
                "FlowerType": {
                    "value": {
                        "interpretedValue": "dozen roses",
                        "originalValue": "dozen roses",
                        "resolvedValues": []
                    }
                },
                "PickupDate": {
                    "value": {
                        "interpretedValue": "2021-01-04",
                        "originalValue": "next monday",
                        "resolvedValues": [
                            "2021-01-04"
                        ]
                    }
                },
                "PickupTime": {
                    "value": {
                        "interpretedValue": "17:00",
                        "originalValue": "5 evening",
```

```
                        "resolvedValues": [
                            "17:00"
                        ]
                    }
                }
            },
            "state": "InProgress"
        },
        "originatingRequestId": "9e8add70-4106-4a10-93f5-2ce2cb959e5f"
    }
}
```

Tourner 5

Au dernier tour, l'utilisateur répond à l'invite de confirmation.

```
{
    "text": "yes"
}
```

Dans la réponse, Amazon Lex V2 envoie des informations indiquant que l'objectif a été atteint en définissant les`confirmationState` valeurs de fin`Confirmed` et`dialogAction` de fin. Toutes les valeurs d'emplacement sont disponibles pour l'application cliente.

```
{
    "interpretations": [
        {
            "intent": {
                "confirmationState": "Confirmed",
                "name": "OrderFlowers",
                "slots": {
                    "FlowerType": {
                        "value": {
                            "interpretedValue": "dozen roses",
                            "originalValue": "dozen roses",
                            "resolvedValues": []
                        }
                    },
                    "PickupDate": {
                        "value": {
```

```
                        "interpretedValue": "2021-01-04",
                        "originalValue": "next monday",
                        "resolvedValues": [
                            "2021-01-04"
                        ]
                    }
                },
                "PickupTime": {
                    "value": {
                        "interpretedValue": "17:00",
                        "originalValue": "5 evening",
                        "resolvedValues": [
                            "17:00"
                        ]
                    }
                }
            },
            "state": "Fulfilled"
        },
        "nluConfidence": {
            "score": 1.0
        }
    },
    {
        "intent": {
            "name": "FallbackIntent",
            "slots": {}
        }
    }
],
"messages": [
    {
        "content": "Thanks. ",
        "contentType": "PlainText"
    }
],
"sessionId": "bf445a49-7165-4fcd-9a9c-a782493fba5c",
"sessionState": {
    "dialogAction": {
        "type": "Close"
    },
    "intent": {
        "confirmationState": "Confirmed",
        "name": "OrderFlowers",
```

```
            "slots": {
                "FlowerType": {
                    "value": {
                        "interpretedValue": "dozen roses",
                        "originalValue": "dozen roses",
                        "resolvedValues": []
                    }
                },
                "PickupDate": {
                    "value": {
                        "interpretedValue": "2021-01-04",
                        "originalValue": "next monday",
                        "resolvedValues": [
                            "2021-01-04"
                        ]
                    }
                },
                "PickupTime": {
                    "value": {
                        "interpretedValue": "17:00",
                        "originalValue": "5 evening",
                        "resolvedValues": [
                            "17:00"
                        ]
                    }
                }
            },
            "state": "Fulfilled"
        },
        "originatingRequestId": "9e8add70-4106-4a10-93f5-2ce2cb959e5f"
    }
}
```

Construire des robots

Vous créez un bot Amazon Lex V2 pour interagir avec vos utilisateurs afin d'obtenir des informations pour accomplir une tâche. Par exemple, vous pouvez créer un bot qui collecte les informations nécessaires pour commander un bouquet de fleurs ou pour réserver une chambre d'hôtel.

Pour créer un bot, vous avez besoin des informations suivantes :

1. Langue utilisée par le bot pour interagir avec le client. Vous pouvez choisir une ou plusieurs langues, chaque langue contenant des intentions, des emplacements et des types d'emplacements indépendants.
2. Les intentions, ou objectifs, que le bot aide l'utilisateur à atteindre. Un bot peut contenir une ou plusieurs intentions, telles que la commande de fleurs ou la réservation d'un hôtel et d'une voiture de location. Vous devez décider quelles déclarations ou quels énoncés l'utilisateur fait pour déclencher l'intention.
3. Les informations, ou créneaux, que vous devez recueillir auprès de l'utilisateur pour répondre à une intention. Par exemple, vous devrez peut-être obtenir le type de fleurs auprès de l'utilisateur ou la date de début d'une réservation d'hôtel. Vous devez définir une ou plusieurs invites utilisées par Amazon Lex V2 pour obtenir la valeur de l'emplacement auprès de l'utilisateur.
4. Type de machines à sous dont vous avez besoin de la part de l'utilisateur. Vous devrez peut-être créer un type de créneau personnalisé, tel qu'une liste de fleurs qu'un utilisateur peut commander, ou vous pouvez utiliser un type de créneau intégré, tel que le type de `AMAZON.Date` créneau pour la date de début d'une réservation.
5. L'interaction de l'utilisateur se déroule au sein des intentions et entre elles. Vous pouvez configurer le flux de conversation pour définir l'interaction entre l'utilisateur et le bot une fois que l'intention est invoquée. Vous pouvez créer une fonction Lambda pour valider et satisfaire l'intention.

Rubriques

> **Note**
>
> Le 17 août 2022, Amazon Lex V2 a publié une modification de la façon dont les conversations sont gérées avec l'utilisateur. Cette modification vous permet de mieux contrôler le chemin emprunté par l'utilisateur tout au long de la conversation. Pour en savoir plus, consultez Comprendre la gestion du flux de conversation. Les robots créés avant le 17 août 2022 ne prennent pas en charge les messages d'accrochage par code de dialogue, la définition de valeurs, la configuration des étapes suivantes et l'ajout de conditions.

Comprendre la gestion du flux de conversation

Le 17 août 2022, Amazon Lex V2 a publié une modification de la façon dont les conversations sont gérées avec l'utilisateur. Cette modification vous permet de mieux contrôler le chemin emprunté par l'utilisateur tout au long de la conversation.

Avant le changement, Amazon Lex V2 gérait la conversation en attribuant des créneaux en fonction de leurs priorités et de leurs intentions. Vous pouvez modifier ce comportement de manière dynamique et modifier le chemin de conversation en fonction des entrées de l'utilisateur DialogAction à l'aide de la fonction Lambda. Pour ce faire, vous pouvez suivre l'état actuel de la conversation et décider par programmation de la marche à suivre en fonction de l'état de la session.

Avec cette modification, vous pouvez créer des chemins de conversation et des branches conditionnelles à l'aide de la console Amazon Lex V2 ou des API sans utiliser de fonction Lambda. Amazon Lex V2 suit l'état de la conversation et contrôle la marche à suivre en fonction des conditions définies lors de la création du bot. Cela vous permet de créer facilement des conversations complexes lors de la conception de votre bot.

Ces modifications vous donnent un contrôle total sur la conversation avec votre client. Toutefois, vous n'êtes pas obligé de définir un chemin. Si vous ne spécifiez pas de chemin de conversation, Amazon Lex V2 crée un chemin par défaut en fonction de la priorité des créneaux que vous souhaitez. Vous pouvez continuer à utiliser les fonctions Lambda pour définir des chemins de conversation de manière dynamique. Dans un tel scénario, la conversation reprend en fonction de l'état de session configuré dans la fonction Lambda.

Cette mise à jour fournit les éléments suivants :

- Une nouvelle expérience de console pour créer des robots avec des flux de conversation complexes.

- Mises à jour des API existantes pour créer des robots afin de prendre en charge les nouveaux flux de conversation.
- Une réponse initiale pour envoyer un message en cas d'invocation intentionnelle.
- Nouvelles réponses pour l'obtention d'un créneau, l'invocation de Lambda en tant que crochet de code de dialogue et la confirmation.
- Possibilité de spécifier les prochaines étapes à chaque étape de la conversation.
- Évaluation des conditions pour concevoir plusieurs parcours de conversation.
- Réglage des valeurs des créneaux et des attributs de session à tout moment de la conversation.

Notez ce qui suit pour les anciens robots :

- Les robots créés avant le 17 août 2022 continuent d'utiliser l'ancien mécanisme pour gérer les flux de conversation. Les robots créés après cette date utilisent la nouvelle méthode de gestion du flux de conversation.
- Les nouveaux robots créés via des importations après le 17 août 2022 utilisent la nouvelle gestion des flux de conversation. Les importations sur des robots existants continuent d'utiliser l'ancienne méthode de gestion des conversations.
- Pour activer la nouvelle gestion du flux de conversation pour un bot créé avant le 17 août 2022, exportez le bot, puis importez-le en utilisant un nouveau nom de bot. Le bot nouvellement créé à partir de l'importation utilise la nouvelle gestion des flux de conversation.

Notez ce qui suit pour les nouveaux robots créés après le 17 août 2022 :

- Amazon Lex V2 suit le flux de conversation défini exactement comme prévu pour fournir l'expérience souhaitée. Vous devez configurer toutes les branches du flux afin d'éviter les chemins de conversation par défaut pendant l'exécution.
- Les étapes de conversation qui suivent un crochet de code doivent être entièrement configurées, car des étapes incomplètes peuvent entraîner l'échec du bot. Nous vous recommandons de valider les robots créés avant le 17 août 2022, car pour ces robots, il n'existe pas de validation automatique des étapes de conversation à la suite d'un crochet de code.

Création d'un robot

Vous pouvez créer un bot avec Amazon Lex V2 de différentes manières :

1. Utilisez la console Amazon Lex V2 pour créer un bot à l'aide d'une interface de site Web. Pour en savoir plus, consultez Création d'un bot à l'aide de la console Amazon Lex V2.
2. Utilisez le Descriptive Bot Builder pour créer un bot à l'aide des fonctionnalités d'intelligence artificielle générative d'Amazon Bedrock. Pour en savoir plus, consultez Utilisation du générateur de bots descriptif.
3. Utilisez des modèles de bot pour créer un bot préconfiguré qui correspond aux cas d'utilisation professionnels courants. Pour en savoir plus, consultez Génération de robots prédéfinis à partir de modèles de robots.
4. Utilisez un AWSSDK pour créer un bot à l'aide d'opérations d'API.
5. Utilisez le concepteur de chatbot automatisé pour créer un bot en utilisant les transcriptions de chat existantes entre les agents et les clients. Pour en savoir plus, consultez Utilisation du concepteur de Chatbot automatisé.
6. Importez une définition de bot existante. Pour en savoir plus, consultez Importation.
7. AWS CloudFormationÀ utiliser pour créer un bot. Pour plus d'informations, consultez Création Amazon Lex V2AWS CloudFormation.

Rubriques

Création d'un bot à l'aide de la console Amazon Lex V2

Commencez à créer votre bot en définissant le nom, la description et certaines informations de base.

Pour créer un bot

1. Connectez-vous à la console Amazon Lex AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/lex/.
2. Choisissez Create bot.
3. Dans la section Méthode de création, choisissez Créer.
4. Dans la section Configuration du bot, attribuez un nom au bot et une description facultative.
5. Dans la section Autorisations IAM, choisissez un rôle AWS Identity and Access Management (IAM) qui fournit à Amazon Lex V2 l'autorisation d'accéder à d'autres AWS services, tels

qu'Amazon. CloudWatch Vous pouvez demander à Amazon Lex V2 de créer le rôle, ou vous pouvez choisir un rôle existant avec CloudWatch des autorisations.

6. Dans la section Loi sur la protection de la vie privée des enfants en ligne (COPPA), choisissez la réponse appropriée.
7. Dans la section Délai d'inactivité de la session, choisissez la durée pendant laquelle Amazon Lex V2 maintient une session ouverte avec un utilisateur. Amazon Lex V2 gère les variables de session pendant toute la durée de la session afin que votre bot puisse reprendre une conversation avec les mêmes variables.
8. Dans la section Paramètres avancés, ajoutez des balises qui permettent d'identifier le bot et peuvent être utilisées pour contrôler l'accès et surveiller les ressources.
9. Choisissez Suivant pour créer le bot et passer à l'ajout d'une langue.

Génération de robots prédéfinis à partir de modèles de robots

Amazon Lex V2 propose des solutions prédéfinies pour créer des expériences à grande échelle et stimuler l'engagement numérique. Les modèles de robots prédéfinis automatisent et normalisent les expériences client. Les modèles de robots fournissent des flux de ready-to-use conversation ainsi que des données d'entraînement et des invites de dialogue, à la fois pour les modalités vocales et de chat. Vous pouvez accélérer la fourniture de solutions robotisées tout en optimisant les ressources, afin de vous concentrer sur les relations avec les clients.

Vous pouvez créer des robots prédéfinis en fonction de votre cas d'utilisation métier. Vous pouvez utiliser la AWS CloudFormation console pour sélectionner les options prédéfinies pour les services associés, tels qu'Amazon S3, Amazon Connect et DynamoDB.

Amazon Lex V2 prend actuellement en charge les secteurs d'activité suivants :

- Services financiers
- Commandes au détail
- Assurance auto
- Télécommunications
- Services aériens
- Plus d'informations à venir prochainement...

Vous pouvez créer un bot à l'aide du modèle de solution métier fourni et le personnaliser en fonction des besoins de votre entreprise.

> **Note**
>
> Les modèles créent des ressources en dehors d'Amazon Lex V2 via des AWS CloudFormation piles. Il peut être nécessaire de modifier la pile dans d'autres consoles telles que Lambda et DynamoDB.

Conditions préalables requises pour créer et déployer le modèle de bot :

- Un compte AWS
- Accès aux AWS services suivants :
  - Amazon Lex V2 pour créer des robots
  - Lambda pour les fonctions de connexion professionnelles
  - DynamoDB pour créer les tables
  - Accès IAM pour créer des politiques et des rôles
  - AWSCloudFormationpour exécuter la pile
- Accès IAM et informations d'identification par clé secrète
- Instance Amazon Connect (facultatif)

> **Note**
>
> L'utilisation de différents AWS services entraîne des coûts d'utilisation respectifs pour chaque service.

Pour créer un bot à partir des modèles Amazon Lex V2 :

1. Connectez-vous à la console Amazon Lex AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/lex/.
2. Sélectionnez le bouton orange qui indique Créer des robots à partir d'un modèle.
3. Sélectionnez le secteur d'activité que vous souhaitez utiliser pour votre modèle de bot. REMARQUE : 5 modèles de robots sont actuellement disponibles. Plus d'informations à venir prochainement.

4. Sélectionnez Créer pour le modèle que vous souhaitez utiliser. Un onglet s'ouvre dans AWS CloudFormation lequel vous pouvez modifier les paramètres de la AWS CloudFormation pile. Toutes les options sont déjà renseignées pour le modèle que vous avez choisi. Vous pouvez également en savoir plus sur le fonctionnement du modèle de bot en sélectionnant En savoir plus.
5. Dans la AWS CloudFormation console, AWS CloudFormation crée une configuration par défaut pour chacune des valeurs du modèle que vous avez choisi. Vous pouvez également sélectionner votre propre nom de pile, vos AWS CloudFormation paramètres, votre table Amazon DynamoDB et (facultatif) les paramètres Amazon Connect.
6. Au bas de la fenêtre, sélectionnez Créer une pile.
7. AWS CloudFormationtraite la demande en arrière-plan pendant plusieurs minutes pour configurer votre nouveau bot. REMARQUE : Le processus crée automatiquement des ressources pour une table DynamoDB, un flux de contacts Amazon Connect et une instance Amazon Connect. Vous pouvez suivre la progression dans la AWS CloudFormation console, puis revenir à la console Amazon Lex V2 une fois la création de la CloudFormation pile terminée.
8. En cas de création réussie, un message s'affiche et vous pouvez sélectionner Accéder à la liste des robots pour accéder à la page des robots, où vous trouverez votre nouveau bot prêt à être testé et utilisé.

Configuration de votre modèle de bot

Fonctions Lambda : le modèle de bot crée automatiquement les fonctions Lambda nécessaires à votre déploiement. Si plusieurs robots font partie de la solution modèle, plusieurs fonctions Lambda sont répertoriées dans les AWS CloudFormation paramètres. Si vous avez des fonctions Lambda existantes à déployer avec votre bot, vous pouvez entrer le nom de votre fonction Lambda personnalisée.

Amazon DynamoDB : le modèle de bot crée automatiquement la table DynamoDB requise pour charger vos exemples de données de politique. Vous pouvez également entrer le nom de votre table DynamoDB personnalisée. Votre table DynamoDB personnalisée doit être formatée de la même manière que la table par défaut créée par le déploiement du modèle de bot.

Amazon Connect — Vous pouvez configurer votre instance Amazon Connect pour qu'elle fonctionne avec votre nouveau modèle de bot en saisissant l'ConnectInstanceARN et un identifiant uniqueContactFlowName. Avec Amazon Connect, vous pouvez tester votre bot à l'aide d'un système IVR de bout en bout.

Résolution des problèmes liés à votre modèle de bot

- Vérifiez que vous disposez des autorisations nécessaires pour créer le modèle que vous choisissez. Les utilisateurs ont besoin de CloudFormation : une CreateStack autorisation ainsi que AWS des autorisations pour les ressources répertoriées dans le modèle. La liste des ressources nécessitant des autorisations utilisateur se trouve au bas de la page Créer un modèle.
- Si votre modèle de bot ne parvient pas à être créé, la bannière rouge de la console Amazon Lex V2 fournit un lien vers la AWS CloudFormation pile responsable de la création du modèle. Dans la AWS CloudFormation console, vous pouvez consulter l'onglet Événements pour voir l'erreur spécifique à l'origine de l'échec du modèle. Une fois que vous avez examiné l'AWS CloudFormationerreur, consultez la section Résolution des problèmes CloudFormation pour plus d'informations.
- Les modèles de robots fonctionnent uniquement avec les exemples de données. Vous devez renseigner le tableau DynamoDB avec vos données pour que les modèles fonctionnent avec vos données personnalisées.

Utilisation du concepteur de Chatbot automatisé

> **Note**
>
> Vous ne pouvez utiliser les transcriptions qu'en anglais (États-Unis).

Le concepteur de chatbot automatisé vous aide à concevoir des robots à partir de transcriptions de conversation existantes. Il analyse les transcriptions et suggère une conception initiale avec des intentions et des types d'emplacements. Vous pouvez modifier la conception du bot, ajouter des invites, créer, tester et déployer le bot.

Après avoir créé un nouveau bot ou ajouté une langue à votre bot à l'aide de la console ou de l'API Amazon Lex V2, vous pouvez télécharger les transcriptions des conversations entre deux parties. Le concepteur de chatbot automatisé analyse les transcriptions et détermine les intentions et les types d'emplacements du bot. Il indique également les conversations qui ont influencé la création d'une intention ou d'un type de créneau particulier pour votre évaluation.

Vous utilisez la console Amazon Lex V2 ou l'API pour analyser les transcriptions de conversation et suggérer des intentions et des types d'emplacements pour un bot.

Vous pouvez consulter les intentions et les types d'emplacements suggérés une fois que le concepteur du chatbot a terminé l'analyse. Après avoir ajouté une intention ou un type d'emplacement suggéré, vous pouvez le modifier ou le supprimer de la conception du bot à l'aide de la console ou de l'API.

Le concepteur de chatbot automatisé prend en charge les fichiers de transcription des conversations à l'aide du schéma Contact Lens for Amazon Connect. Si vous utilisez une autre application de centre de contact, vous devez transformer les transcriptions de conversation au format utilisé par le concepteur du chatbot. Pour plus d'informations, consultez Format de transcription d'entrée.

Pour utiliser le concepteur de chatbot automatisé, vous devez autoriser le rôle IAM qui exécute le concepteur à accéder. Pour la politique IAM spécifique, voirAutoriser les utilisateurs à utiliser le concepteur de Chatbot automatisé. Pour permettre à Amazon Lex V2 de chiffrer les données de sortie à l'aide d'une AWS KMS clé facultative, vous devez mettre à jour la clé conformément à la politique indiquée dansAutoriser les utilisateurs à utiliser une AWS KMS clé pour chiffrer et déchiffrer des fichiers.

> **Note**
>
> Si vous utilisez unKMS key, vous devez fournir une KMS keypolitique, quel que soit le IAM rôle utilisé.

Rubriques

- Importation de transcriptions de conversation
- Création d'intentions et de types d'emplacements
- Format de transcription d'entrée
- Format de transcription de sortie

Importation de transcriptions de conversation

L'importation des transcriptions de conversation se fait en trois étapes :

1. Préparez les transcriptions pour l'importation en les convertissant au format approprié. Si vous utilisez Contact Lens pour Amazon Connect, les transcriptions sont déjà au bon format.
2. Téléchargez les transcriptions dans un compartiment Amazon S3. Si vous utilisez Contact Lens, vos transcriptions se trouvent déjà dans un compartiment S3.

3. Analysez les transcriptions à l'aide de la console Amazon Lex V2 ou des opérations d'API.
   Le temps nécessaire pour terminer la formation dépend du volume de transcriptions et de la complexité de la conversation. En général, 500 lignes de transcriptions sont analysées chaque minute.

Chacune de ces étapes est décrite dans les sections suivantes.

Importation de transcriptions depuis Contact Lens pour Amazon Connect

Le concepteur de chatbot automatisé Amazon Lex V2 est compatible avec les fichiers de transcription des lentilles de contact. Pour utiliser les fichiers de transcription de Contact Lens, vous devez activer Contact Lens et noter l'emplacement de ses fichiers de sortie.

Pour exporter des transcriptions depuis Contact Lens

1. Activez Contact Lens dans votre instance Amazon Connect. Pour obtenir des instructions, consultez la section Activer les lentilles de contact pour Amazon Connect dans le guide de l'administrateur Amazon Connect.
2. Notez l'emplacement du compartiment S3 qu'Amazon Connect utilise pour votre instance. Pour connaître l'emplacement, ouvrez la page Stockage des données dans la console Amazon Connect. Pour obtenir des instructions, consultez la section Mettre à jour les paramètres de l'instance dans le guide de l'administrateur Amazon Connect.

Après avoir activé les lentilles de contact et noté l'emplacement de vos fichiers de transcription, consultez les instructions Analysez vos transcriptions à l'aide de la console Amazon Lex V2 pour importer et analyser vos transcriptions.

Préparer les transcriptions

Préparez vos transcriptions en créant des fichiers de transcription.

- Créez un fichier de transcription par conversation répertoriant les interactions entre les parties. Chaque interaction de la conversation peut s'étendre sur plusieurs lignes. Vous pouvez fournir des versions expurgées et non expurgées de la conversation.
- Le fichier doit être au format JSON spécifié dansFormat de transcription d'entrée.
- Vous devez fournir au moins 1 000 tours de conversation. Pour améliorer la découverte de vos intentions et de vos types de machines à sous, vous devez prévoir environ 10 000 tours de

conversation ou plus. Le concepteur de chatbot automatisé ne traitera que les 700 000 premiers tours.

- Il n'y a pas de limite au nombre de fichiers de transcription que vous pouvez télécharger, pas plus qu'il n'y a de restriction de taille de fichier.

Si vous prévoyez de filtrer les transcriptions que vous importez par date, les fichiers doivent se trouver dans la structure de répertoires suivante :

```
<path or bucket root>
   --> yyyy
      --> mm
         --> dd
            --> transcript files
```

Le fichier de transcription doit contenir la date au format « yyyy-mm-dd » quelque part dans le nom du fichier.

Pour exporter des transcriptions depuis d'autres applications de centre de contact

1. Utilisez les outils de votre application de centre d'appels pour exporter les conversations. La conversation doit contenir au moins les informations spécifiées dansFormat de transcription d'entrée.
2. Transformez les transcriptions produites par votre application de centre de contact au format décrit dansFormat de transcription d'entrée. Vous êtes responsable de la réalisation de la transformation.

Nous fournissons trois scripts pour préparer les transcriptions. Il s'agit des options suivantes :

- Script permettant de combiner les transcriptions de lentilles de contact avec les journaux de conversation Amazon Lex V2. Les transcriptions de lentilles de contact n'incluent pas les parties des conversations Amazon Connect qui interagissent avec les robots Amazon Lex V2. Le script nécessite l'activation des journaux de conversation pour Amazon Lex V2 et les autorisations appropriées pour interroger les CloudWatch journaux des conversations et les compartiments Contact Lens S3.
- Un script pour transformer les analyses d'appels Amazon Transcribe au format d'entrée Amazon Lex V2.

- Un script pour transformer les transcriptions de chat Amazon Connect au format d'entrée Amazon Lex V2.

Vous pouvez télécharger les scripts depuis ce GitHub dépôt : https://github.com/aws-samples/amazon-lex-bot-recommendation -integration.

Téléchargez vos transcriptions dans un compartiment S3

Si vous utilisez Contact Lens, vos fichiers de transcription sont déjà contenus dans un compartiment S3. Pour connaître l'emplacement et les noms de fichiers de vos transcriptions, consultez la section Exemples de fichiers de sortie pour lentilles de contact dans le guide de l'administrateur Amazon Connect.

Si vous utilisez une autre application de centre de contact et que vous n'avez pas configuré de compartiment S3 pour vos fichiers de transcription, suivez cette procédure. Sinon, si vous possédez déjà un compartiment S3, après vous être connecté à la console Amazon S3, suivez cette procédure en commençant par l'étape 5.

Pour charger les fichiers dans un compartiment S3

1. Connectez-vous à la AWS Management Console et ouvrez la console Simple Storage Service (Amazon S3) à la page https://console.aws.amazon.com/s3/.
2. Choisissez Créer un compartiment.
3. Donnez un nom au compartiment et choisissez une région. La région doit être la même que celle que vous utilisez pour Amazon Lex V2. Définissez les autres options selon les besoins de votre cas d'utilisation.
4. Choisissez Create bucket (Créer un compartiment).
5. Dans la liste des buckets, choisissez un bucket existant ou le bucket que vous venez de créer
6. Sélectionnez Charger.
7. Ajoutez les fichiers de transcription que vous souhaitez télécharger.
8. Sélectionnez Charger.

Analysez vos transcriptions à l'aide de la console Amazon Lex V2

Vous ne pouvez utiliser la conception automatique de robots que dans une langue vide. Vous pouvez ajouter une nouvelle langue à un bot existant ou en créer un nouveau.

Pour créer une nouvelle langue dans un nouveau bot

1. Connectez-vous à la console Amazon Lex AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/lex/.
2. Choisissez Create bot
3. Choisissez Démarrer avec Automated Chatbot Designer. Renseignez les informations pour créer votre nouveau bot.
4. Choisissez Next (Suivant)
5. Dans Ajouter une langue pour le bot, renseignez les informations relatives à la langue.
6. Dans la section Emplacement du fichier de transcription sur S3, choisissez le compartiment S3 qui contient vos fichiers de transcription et le chemin local vers les fichiers si nécessaire.
7. Vous pouvez éventuellement choisir les options suivantes :

   - Une AWS KMS clé pour chiffrer les données de transcription pendant le traitement. Si vous ne sélectionnez aucune clé, une AWS KMS clé de service est utilisée.
   - Pour filtrer les transcriptions selon une plage de dates spécifique. Si vous choisissez de filtrer les transcriptions, elles doivent se trouver dans la bonne structure de dossiers. Pour en savoir plus, consultez Préparer les transcriptions.
8. Sélectionnez Exécuté.

Attendez qu'Amazon Lex V2 traite la transcription. Un message de fin d'analyse s'affiche lorsque l'analyse est terminée.

Comment arrêter d'analyser votre relevé de notes

Si vous devez arrêter l'analyse des transcriptions que vous avez téléchargées, vous pouvez arrêter une `BotRecommendation` tâche en cours, qui a le `BotRecommendationStatus` statut de traitement. Vous pouvez cliquer sur le bouton Arrêter le traitement présent sur la bannière après avoir soumis une tâche depuis la console ou en utilisant le SDK CLI pour l'`StopBotRecommendation`API. Pour plus d'informations, voir StopBotRecommendation

Après avoir appelé le`StopBotRecommendation`, le système interne `BotRecommendationStatus` est réglé sur `Stopping` et vous n'êtes pas débité. Pour vous assurer que le travail s'est arrêté, vous pouvez appeler l'`DescribeBotRecommendation`API et vérifier que `BotRecommendationStatus` c'est le cas`Stopped`. Cela prend généralement 3 à 4 minutes.

Le traitement ne vous est pas facturé après l'appel de l'`StopBotRecommendation`API.

Création d'intentions et de types d'emplacements

Une fois que le concepteur du chatbot a créé les intentions et les types d'emplacements, vous sélectionnez les intentions et les types d'emplacements à ajouter à votre bot. Vous pouvez consulter les détails de chaque intention et de chaque type d'emplacement pour vous aider à choisir les recommandations les plus pertinentes pour votre cas d'utilisation.

Vous pouvez cliquer sur le nom d'une intention recommandée pour afficher les exemples d'énoncés et de créneaux suggérés par le concepteur du chatbot. Si vous sélectionnez Afficher les transcriptions associées, vous pouvez également faire défiler les conversations que vous avez fournies. Ces transcriptions influencent la recommandation du concepteur du chatbot quant à cette intention. Si vous cliquez sur un exemple d'énoncé, vous pouvez consulter la conversation principale et la tournure de dialogue correspondante, qui ont influencé cet énoncé spécifique.

Vous pouvez cliquer sur le nom d'un type d'emplacement spécifique pour afficher les valeurs d'emplacement recommandées. Si vous sélectionnez Afficher les transcriptions associées, vous pouvez consulter les conversations qui ont influencé ce type de créneau, en surlignant l'invite de l'agent demandant le type de créneau. Si vous cliquez sur une valeur de type de slot spécifique, vous pouvez consulter la conversation principale et la tournure de dialogue correspondante qui ont influencé cette valeur.

Pour consulter et ajouter des intentions et un type d'emplacement

1. Connectez-vous à la console Amazon Lex AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/lex/.
2. Dans la liste des robots, choisissez celui avec lequel vous souhaitez travailler.
3. Choisissez Afficher les langues.
4. Dans la liste des langues, choisissez la langue avec laquelle vous souhaitez travailler.
5. Dans Structure de conversation, sélectionnez Réviser.
6. Dans la liste des intentions et des types d'emplacements, choisissez ceux à ajouter au bot. Vous pouvez choisir une intention ou un type de créneau pour voir les détails et les transcriptions associées.

Les intentions sont triées en fonction du niveau de confiance d'Amazon Lex V2 quant au fait que l'intention est associée aux transcriptions traitées.

Format de transcription d'entrée

Le format de fichier d'entrée suivant permet de générer les intentions et les types d'emplacements pour votre bot. Le fichier d'entrée doit contenir ces champs. Les autres champs sont ignorés.

Le format d'entrée est compatible avec le format de sortie de Contact Lens pour Amazon Connect. Si vous utilisez Contact Lens, vous n'avez pas besoin de modifier vos fichiers de transcription. Pour plus d'informations, consultez la section Exemples de fichiers de sortie pour lentilles de contact. Si vous utilisez une autre application de centre de contact, vous devez transformer votre fichier de transcription dans ce format.

```
{
    "Participants": [
        {
            "ParticipantId": "string",
            "ParticipantRole": "AGENT | CUSTOMER"
        }
    ],
    "Version": "1.1.0",
    "ContentMetadata": {
        "RedactionTypes": [
            "PII"
        ],
        "Output": "Raw | Redacted"
    },
    "CustomerMetadata": {
        "ContactId": "string"
    },
    "Transcript": [
        {
            "ParticipantId": "string",
            "Id": "string",
            "Content": "string"
        }
    ]
}
```

Les champs suivants doivent être présents dans le fichier d'entrée :

- Participants Identifie les participants à la conversation et le rôle qu'ils jouent.
- Version Version du format de fichier d'entrée. Toujours « 1.1.0 ».

- ContentMetadataIndique si vous avez supprimé des informations sensibles de la transcription. Définissez le `Output` champ sur « Raw » si la transcription contient des informations sensibles.
- CustomerMetadataIdentifiant unique pour la conversation.
- Transcription Le texte de la conversation entre les parties impliquées dans la conversation. Chaque tour de conversation est identifié par un identifiant unique.

Format de transcription de sortie

Le format de transcription de sortie est presque identique au format de transcription d'entrée. Cependant, il inclut également certaines métadonnées relatives aux clients et un champ répertoriant les segments qui ont influencé la suggestion des intentions et des types de créneaux. Vous pouvez télécharger la transcription de sortie depuis la page de révision de la console ou à l'aide de l'API Amazon Lex V2. Pour en savoir plus, consultez Format de transcription d'entrée.

```
{
    "Participants": [
        {
            "ParticipantId": "string",
            "ParticipantRole": "AGENT | CUSTOMER"
        }
    ],
    "Version": "1.1.0",
    "ContentMetadata": {

        "RedactionTypes": [
            "PII"
        ],
        "Output": "Raw | Redacted"
    },
    "CustomerMetadata": {
        "ContactId": "string",
        "FileName": "string",
        "InputFormat": "Lex"
    },
    "InfluencingSegments": [
        {
            "Id": "string",
            "StartTurnIndex": number,
            "EndTurnIndex": number,
            "Intents": [
                {
```

```
                    "Id": "string",
                    "Name": "string",
                    "SampleUtteranceIndex": [
                        {
                            "Index": number,
                            "Content": "String"
                        }
                    ]
                }
            ],
            "SlotTypes": [
                {
                    "Id": "string",
                    "Name": "string",
                    "SlotValueIndex": [
                        {
                            "Index": number,
                            "Content": "String"
                        }
                    ]
                }
            ]
        }
    ],
    "Transcript": [
        {

            "ParticipantId": "string",
            "Id": "string",
            "Content": "string"
        }
    ]
}
```

- CustomerMetadata— Deux champs ont été ajoutés au `CustomerMetadata` champ, le nom du fichier d'entrée contenant la conversation et le format de saisie, qui est toujours « Lex ».
- InfluencingSegments— Identifie les segments de la conversation qui ont influencé la suggestion d'une intention ou d'un type de créneau. L'identifiant de l'intention ou du type de créneau identifie la personne spécifique influencée par la conversation.

Ajouter une langue

Vous ajoutez une ou plusieurs langues et paramètres régionaux à votre bot pour lui permettre de communiquer avec les utilisateurs dans leur langue. Vous définissez les intentions, les créneaux et les types d'emplacements séparément pour chaque langue afin que les valeurs des énoncés, des invites et des emplacements soient spécifiques à la langue.

Votre bot doit contenir au moins une langue.

Pour ajouter une langue à votre bot

1. Dans la section Nouvelle langue, choisissez la langue que vous souhaitez utiliser. Vous pouvez ajouter une description pour aider à identifier la langue dans les listes.
2. Si votre bot prend en charge l'interaction vocale, dans la section Interaction vocale, choisissez la voix Amazon Polly qu'Amazon Lex V2 utilise pour communiquer avec l'utilisateur. Si votre bot ne prend pas en charge la voix, choisissez Aucun.
3. Pour le seuil de confiance de la classification des intentions, définissez la valeur qu'Amazon Lex V2 utilise pour déterminer si une intention est la bonne. Vous pouvez ajuster cette valeur après avoir testé votre bot.
4. Choisissez Add (Ajouter).

Ajouter des intentions

Les intentions sont les objectifs que vos utilisateurs souhaitent atteindre, tels que la commande de fleurs ou la réservation d'un hôtel. Votre bot doit avoir au moins une intention.

Par défaut, tous les robots contiennent une seule intention intégrée, l'intention de repli. Cette intention est utilisée lorsqu'Amazon Lex V2 ne reconnaît aucune autre intention. Par exemple, si un utilisateur déclare « Je souhaite commander des fleurs » dans le cadre d'une intention de réservation d'hôtel, l'intention de repli est déclenchée.

Pour ajouter une intention

1. Connectez-vous auAWS Management Consoleet ouvrez la console Amazon Lex à l'adressehttps://console.aws.amazon.com/lex/.
2. Dans la liste des robots, choisissez le bot auquel vous souhaitez ajouter l'intention, puis depuisAjouter des langueschoisirAfficher les langues.

3. Choisissez la langue à laquelle ajouter l'intention, puis choisissezIntentions.
4. ChoisissezAjouter une intention, donnez un nom à votre intention, puis choisissezAjouter.
5. Dans l'éditeur d'intention, ajoutez les détails de votre intention.

   - Flux de conversation— Utilisez le diagramme de flux de conversation pour voir à quoi pourrait ressembler un dialogue avec votre bot. Vous pouvez choisir différentes sections de la conversation pour accéder à cette section de l'éditeur d'intention.
   - Détails de l'intention— Donnez un nom et une description à l'intention pour aider à identifier le but de l'intention. Vous pouvez également voir l'identifiant unique qu'Amazon Lex V2 a attribué à l'intention.
   - Contextes— Définissez les contextes d'entrée et de sortie correspondant à l'intention. Un contexte est une variable d'état associée à une intention. Un contexte de sortie est défini lorsqu'une intention est remplie. Une intention associée à un contexte de saisie ne peut être reconnue que si le contexte est actif. Une intention sans contexte de saisie peut toujours être reconnue.
   - Exemples d'énoncés— Vous devez fournir au moins 10 phrases que vous vous attendez à ce que vos utilisateurs utilisent pour initier une intention. Amazon Lex V2 généralise à partir de ces phrases pour reconnaître que c'est l'utilisateur qui souhaite initier l'intention.
   - Réponse initiale— Le message initial envoyé à l'utilisateur après l'appel de l'intention. Vous pouvez fournir des réponses, initialiser des valeurs et définir l'étape suivante qu'Amazon Lex V2 doit suivre pour répondre à l'utilisateur dès le début de l'intention.
   - Machines à sous— Définissez les créneaux, ou les paramètres, nécessaires pour atteindre l'objectif. Chaque slot possède un type qui définit les valeurs qui peuvent être saisies dans le slot. Vous pouvez choisir parmi vos types d'emplacement personnalisés ou vous pouvez choisir un type d'emplacement intégré.
   - Confirmation— Ces invites et réponses sont utilisées pour confirmer ou refuser la réalisation de l'intention. L'invite de confirmation demande à l'utilisateur de vérifier les valeurs des créneaux. Par exemple, « J'ai réservé une chambre d'hôtel pour vendredi. Est-ce correct ? » La réponse de refus est envoyée à l'utilisateur lorsqu'il refuse la confirmation. Vous pouvez fournir des réponses, définir des valeurs et définir l'étape suivante qu'Amazon Lex V2 doit suivre en cas de réponse de confirmation ou de refus de l'utilisateur.
   - Exécution— Réponse envoyée à l'utilisateur en cours de traitement. Vous pouvez définir des mises à jour de l'état d'avancement du traitement au début du traitement et régulièrement pendant que celui-ci est en cours d'exécution. Par exemple, « Je suis en train de modifier votre mot de passe, cela peut prendre quelques minutes » et « Je travaille toujours sur votre

demande ». Les mises à jour de Fulfillment ne sont utilisées que pour les conversations en streaming. Vous pouvez également définir un message de réussite après l'exécution, un message d'échec et un message de délai d'expiration. Vous pouvez envoyer des messages après l'expédition, à la fois pour le streaming et pour des conversations régulières. Par exemple, si le traitement aboutit, vous pouvez envoyer « J'ai changé votre mot de passe ». Si le traitement échoue, vous pouvez envoyer une réponse contenant plus d'informations, par exemple « Je n'ai pas pu modifier votre mot de passe, contactez le service d'assistance pour obtenir de l'aide ». Si le traitement prend plus de temps que le délai d'expiration configuré, vous pouvez envoyer un message d'information à l'utilisateur, tel que « Nos serveurs sont très occupés en ce moment. Réessayez votre demande ultérieurement. » Vous pouvez fournir des réponses, définir des valeurs et définir l'étape suivante qu'Amazon Lex V2 doit suivre pour répondre à l'utilisateur.

- Réponses finales— Réponse envoyée à l'utilisateur une fois que l'intention est satisfaite et que tous les autres messages sont lus. Par exemple, un remerciement pour la réservation d'une chambre d'hôtel. Il peut également inciter l'utilisateur à formuler une intention différente, par exemple : « Merci d'avoir réservé une chambre, souhaitez-vous réserver une voiture de location ? » Vous pouvez fournir des réponses et configurer les prochaines actions de suivi après avoir atteint l'objectif et répondu par la réponse finale.
- Crochets de code— Indiquez si vous utilisezAWS Lambdafonction permettant d'initialiser l'intention et de valider la saisie de l'utilisateur. Vous spécifiez la fonction Lambda dans l'alias que vous utilisez pour exécuter le bot.

6. ChoisissezEnregistrer l'intentionpour enregistrer l'intention.

> **Note**
>
> Le 17 août 2022, Amazon Lex V2 a publié une modification de la façon dont les conversations sont gérées avec l'utilisateur. Cette modification vous permet de mieux contrôler le chemin emprunté par l'utilisateur tout au long de la conversation. Pour plus d'informations, veuillez consulter Comprendre la gestion du flux de conversation. Les robots créés avant le 17 août 2022 ne prennent pas en charge les messages d'accroche à code de dialogue, la définition de valeurs, la configuration des étapes suivantes et l'ajout de conditions.

Configuration des invites dans un ordre spécifique

Vous pouvez configurer le bot pour qu'il lise les messages dans un ordre prédéfini en cochant la caseLire les messages dans l'ordre. Sinon, le bot lit le message et les variations dans un ordre aléatoire.

Les invites ordonnées permettent de lire le message et les variantes d'un groupe de messages dans l'ordre des tentatives. Vous pouvez utiliser une autre formulation d'un message lorsque l'utilisateur donne une réponse non valide à l'invite ou pour confirmer son intention. Jusqu'à deux variantes du message d'origine peuvent être définies dans chaque emplacement. Vous pouvez choisir de lire les messages dans l'ordre ou de manière aléatoire.

Ordered prompt prend en charge les quatre types de messages : texte, réponse de charge utile personnalisée, SSML et groupe de cartes. Les réponses sont classées dans le même groupe de messages. Les différents groupes de messages sont indépendants.

Rubriques

- Exemples d'énoncés
- Structure de l'intention
- Création de parcours de conversation
- Utilisation du générateur de conversation visuel
- Intentions prédéfinies

Exemples d'énoncés

Vous créez des exemples d'énoncés qui sont des variantes de phrases que vous vous attendez à ce que les utilisateurs utilisent pour initier une intention. Par exemple, pour une **`BookFlight`** intention, vous pouvez inclure des énoncés tels que les suivants :

1. Je souhaite réserver un vol
2. aidez-moi à prendre l'avion.
3. des billets d'avion, s'il vous plaît !
4. vol de {*`DepartureCity`*} à {*`DestinationCity`*}

Vous devez fournir au moins 10 exemples d'énoncés. Donnez des exemples représentant un large éventail de structures de phrases et de mots que les utilisateurs peuvent prononcer. Pensez

également aux phrases incomplètes, comme dans les exemples 3 et 4 ci-dessus. Vous pouvez également utiliser des espaces que vous avez définis à cet effet dans un exemple d'énoncé en plaçant le nom de l'intervalle entre accolades, comme dans {*DepartureCity*} dans l'exemple 4. Si vous incluez des noms de slot dans un exemple d'énoncé, Amazon Lex V2 remplit les cases correspondant à l'intention avec les valeurs que l'utilisateur fournit dans l'énoncé.

Une variété d'exemples d'énoncés permet à Amazon Lex V2 de généraliser afin de reconnaître efficacement que l'utilisateur souhaite initier l'intention.

Vous pouvez ajouter des exemples d'énoncés dans l'éditeur d'intention, le générateur de conversation visuel ou à l'aide des opérations de l'UpdateIntentAPI CreateIntentou. Vous pouvez également générer automatiquement des exemples d'énoncés en tirant parti des fonctionnalités d'intelligence artificielle générative d'Amazon Bedrock. Pour en savoir plus, consultez Génération d'énoncés.

Utilisez l'éditeur Intent ou le générateur de conversation visuel

1. Dans l'éditeur Intent, accédez à la section Exemples d'énoncés. Dans le générateur de conversation visuel, recherchez la section Exemples d'énoncés dans le bloc Démarrer.
2. Dans la zone contenant le texte transparent**`I want to book a flight`**, tapez un exemple d'énoncé. Sélectionnez Ajouter un énoncé pour ajouter l'énoncé.
3. Consultez les exemples d'énoncés que vous avez ajoutés en mode Aperçu ou en mode texte brut. En texte brut, chaque ligne est un énoncé distinct. En mode aperçu, passez le pointeur de la souris sur un énoncé pour afficher les options suivantes :
   - Sélectionnez la zone de texte pour modifier l'énoncé.
   - Cliquez sur le bouton X à droite de la zone de texte pour supprimer l'énoncé.
   - Faites glisser le bouton situé à gauche de la zone de texte pour modifier l'ordre des exemples d'énoncés.
4. Utilisez la barre de recherche en haut pour effectuer une recherche parmi vos exemples d'énoncés et le menu déroulant situé à côté pour les trier selon l'ordre dans lequel vous les avez ajoutés ou par ordre alphabétique.

Utiliser une opération d'API

1. Créez une nouvelle intention avec l'CreateIntentopération ou mettez à jour une intention existante avec l'UpdateIntentopération.

2. La demande d'API inclut un `sampleUtterances` champ qui correspond à un tableau d'SampleUtteranceobjets.
3. Pour chaque exemple d'énoncé que vous souhaitez ajouter, ajoutez un `SampleUtterance` objet au tableau. Ajoutez l'extrait d'énoncé comme valeur du `utterance` champ.
4. Pour modifier et supprimer des exemples d'énoncés, envoyez une `UpdateIntent` demande. La liste des énoncés que vous fournissez dans le `sampleUtterances` champ remplace les énoncés existants.

> **Important**
>
> Tout champ que vous laissez vide dans la `UpdateIntent` demande entraînera la suppression des configurations existantes dans l'intention de le faire. Utilisez cette DescribeIntentopération pour renvoyer la configuration du bot et copiez les configurations que vous ne souhaitez pas supprimer dans la `UpdateIntent` demande.

Structure de l'intention

Les rubriques suivantes décrivent les différentes étapes qu'un bot doit suivre pour réaliser une intention et comment configurer chacune de ces étapes :

Rubriques

- Réponse initiale
- Emplacements
- Confirmation
- Exécution
- Réponse finale

Réponse initiale

La réponse initiale est envoyée à l'utilisateur une fois qu'Amazon Lex V2 a déterminé l'intention et avant qu'il ne commence à obtenir des valeurs d'emplacement. Vous pouvez utiliser cette réponse pour informer l'utilisateur de l'intention qui a été reconnue et pour le préparer aux informations que vous collectez pour atteindre l'intention.

Par exemple, si l'intention est de planifier un rendez-vous d'entretien pour une voiture, la réponse initiale peut être la suivante :

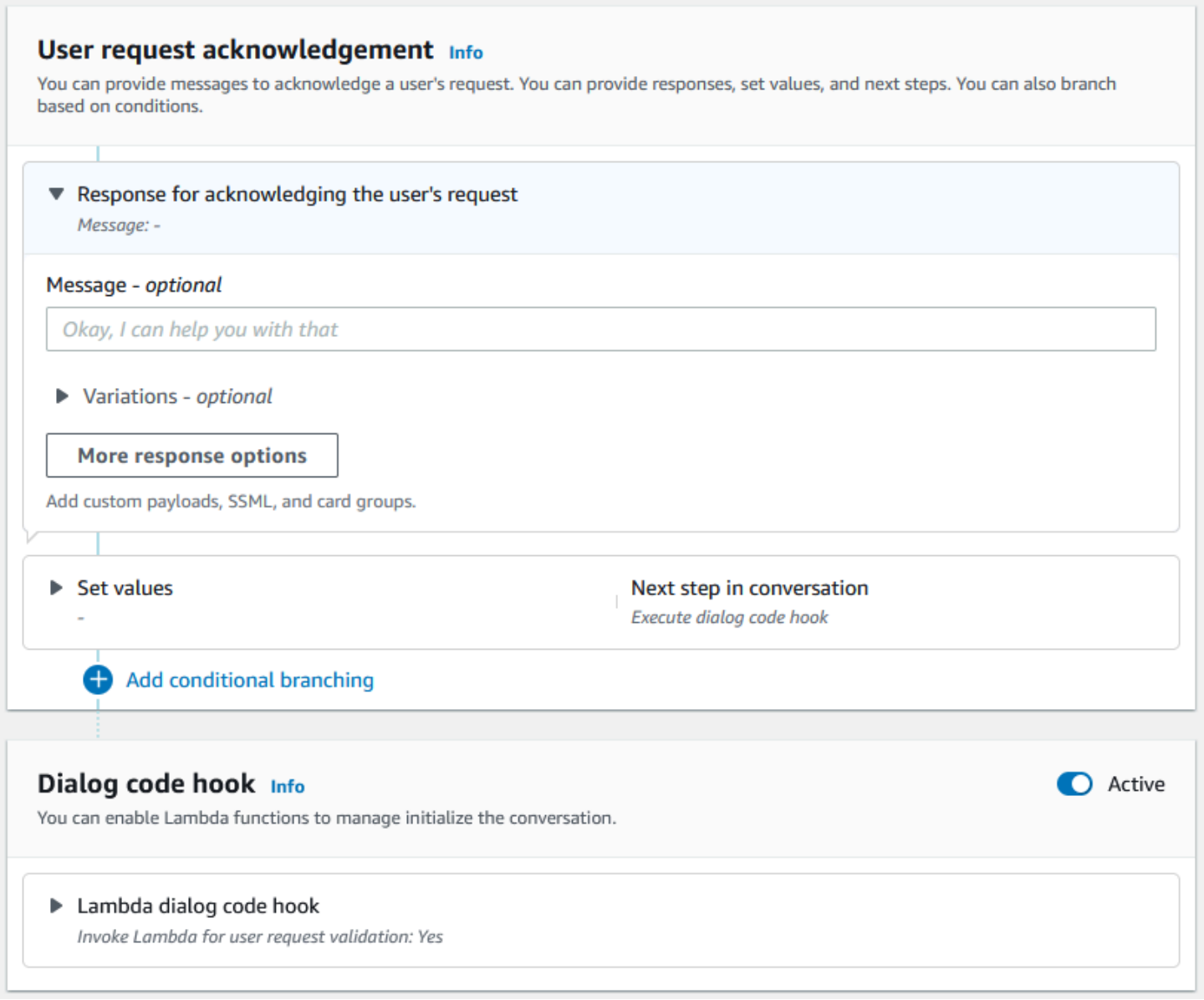
> Je peux vous aider à fixer un rendez-vous. Vous devrez fournir la marque, le modèle et l'année de votre voiture.

Un message de réponse initial n'est pas requis. Si vous n'en fournissez pas, Amazon Lex V2 continue de suivre l'étape suivante de la réponse initiale.

Vous pouvez configurer les options suivantes dans la réponse initiale :

- Configuration de l'étape suivante— Vous pouvez définir l'étape suivante de la conversation, par exemple passer à une action de dialogue spécifique, obtenir un créneau particulier ou passer à une autre intention. Pour plus d'informations, veuillez consulter Configurer les prochaines étapes de la conversation.
- Définir des valeurs— Vous pouvez définir des valeurs pour les créneaux et les attributs de session. Pour de plus amples informations, consultez Définissez des valeurs au cours de la conversation.
- Ajouter un branchement conditionnel— Vous pouvez appliquer des conditions après avoir joué la réponse initiale. Lorsqu'une condition est évaluée comme vraie, les actions que vous définissez sont prises. Pour plus d'informations, veuillez consulter Ajouter des conditions aux conversations dans les succursales.
- Exécuter un crochet de code de dialogue— Vous pouvez définir un hook de code Lambda pour initialiser les données et exécuter la logique métier. Pour plus d'informations, veuillez consulter Invoquer le crochet de code de dialogue. Si l'option permettant d'exécuter la fonction Lambda est activée pour l'intention, le hook du code de dialogue est exécuté par défaut. Vous pouvez désactiver le crochet de code de boîte de dialogue en activant leActifbouton.

En l'absence d'une condition ou d'une étape suivante explicite, Amazon Lex V2 passe au créneau suivant par ordre de priorité.



> **Note**
>
> Le 17 août 2022, Amazon Lex V2 a publié une modification de la façon dont les conversations sont gérées avec l'utilisateur. Cette modification vous permet de mieux contrôler le chemin emprunté par l'utilisateur tout au long de la conversation. Pour plus d'informations, veuillez consulter Comprendre la gestion du flux de conversation. Les robots créés avant le 17 août 2022 ne prennent pas en charge les messages d'accroche à code de dialogue, la définition de valeurs, la configuration des étapes suivantes et l'ajout de conditions.

Emplacements

Les créneaux sont des valeurs fournies par l'utilisateur pour répondre à l'intention. Il existe deux types de machines à sous :

- Type de fente intégré— Vous pouvez utiliser des types d'emplacements intégrés pour capturer des valeurs standard telles que le numéro, le nom et la ville. Pour obtenir la liste des types d'emplacements intégrés pris en charge, voirTypes de slots intégrés.
- Type de slot personnalisé— Vous pouvez utiliser des types d'emplacements personnalisés pour capturer des valeurs personnalisées spécifiques à l'intention. Par exemple, vous pouvez utiliser un type de créneau personnalisé pour capturer le type de compte « Chèque » ou « Épargne ». Pour plus d'informations, veuillez consulter Type de slot personnalisé.

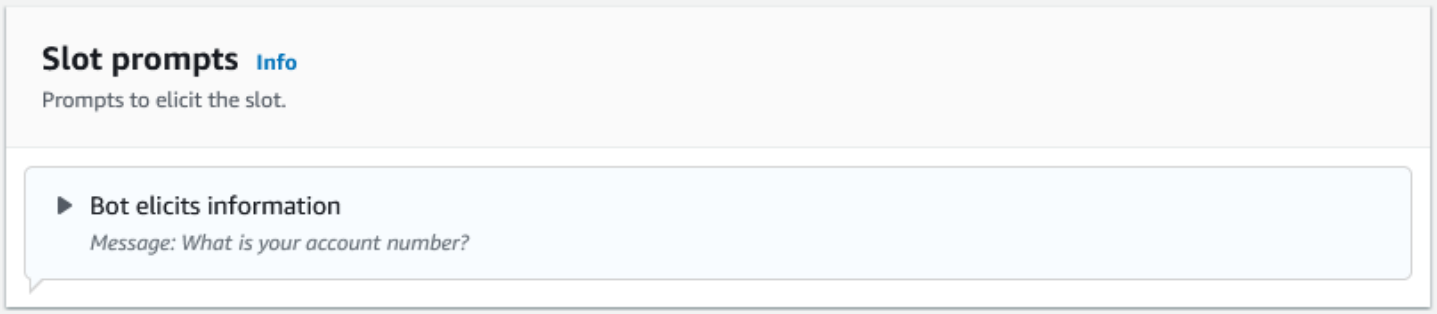
Pour définir un créneau dans une intention, vous devez configurer les éléments suivants :

- Informations sur le slot— Ce champ contient un nom et une description facultative pour l'emplacement. Par exemple, vous pouvez fournir le nom de l'emplacement sous la forme »AccountNumber» pour saisir les numéros de compte. Si le créneau est requis dans le cadre du flux de conversation pour répondre à l'intention, il doit être marqué comme étant obligatoire.
- Type de fente— Un type de slot définit la liste des valeurs qu'un slot peut accepter. Vous pouvez créer un type d'emplacement personnalisé ou utiliser un type d'emplacement prédéfini.
- Slot prompt— Un créneau est une question posée à l'utilisateur pour recueillir des informations. Vous pouvez configurer le nombre de tentatives utilisées pour recueillir des informations et la variation de l'invite utilisée pour chaque nouvelle tentative. Vous pouvez également activer l'invocation d'une fonction Lambda après chaque nouvelle tentative pour traiter l'entrée capturée et tenter de la résoudre en une entrée valide.
- Attendre et continuer (facultatif)— En activant ce comportement, les utilisateurs peuvent prononcer des phrases telles que « attendez une seconde » pour que le bot attende qu'ils trouvent les informations et les fournissent. Ceci n'est activé que pour les conversations en streaming. Pour plus d'informations, veuillez consulter Permettre au bot d'attendre que l'utilisateur fournisse plus d'informations.
- Réponses de capture d'emplacements— Vous pouvez configurer une réponse de réussite et une réponse d'échec en fonction du résultat de la capture de la valeur du slot à partir des données saisies par l'utilisateur.
- Branchement conditionnel— Vous pouvez appliquer des conditions après avoir joué la réponse initiale. Lorsqu'une condition est évaluée comme vraie, les actions que vous définissez sont prises.

Pour plus d'informations, veuillez consulter Ajouter des conditions aux conversations dans les succursales.

- Crochet de code de dialogue— Vous pouvez également utiliser un hook de code Lambda pour valider les valeurs des emplacements et exécuter la logique métier. Pour plus d'informations, veuillez consulter Invoquer le crochet de code de dialogue.
- Type d'entrée utilisateur— Vous pouvez configurer le type d'entrée afin que le bot puisse accepter une modalité spécifique. Par défaut, les modalités audio et DTMF sont acceptées. Vous pouvez le régler de manière sélective sur audio uniquement ou sur DTMF uniquement.
- Délais et durées d'entrée audio— Vous pouvez configurer les délais d'expiration audio, y compris le délai d'expiration de la voix et le délai d'expiration du silence. Vous pouvez également définir la longueur audio maximale.
- Délai d'entrée DTMF, caractères et longueurs— Vous pouvez définir le délai d'expiration du DTMF ainsi que le caractère de suppression et le caractère de fin. Vous pouvez également définir la longueur maximale du DTMF.
- Longueur du texte— Vous pouvez définir la longueur maximale pour la modalité du texte.

Une fois que l'invite de machine à sous est jouée, l'utilisateur fournit la valeur de la machine à sous forme d'entrée. Si Amazon Lex V2 ne comprend pas la valeur d'un emplacement fournie par l'utilisateur, il essaie à nouveau d'obtenir l'emplacement jusqu'à ce qu'il comprenne une valeur ou qu'il dépasse le nombre maximum de nouvelles tentatives que vous avez configuré pour l'emplacement. À l'aide des paramètres avancés des nouvelles tentatives, vous pouvez configurer les délais d'expiration, restreindre le type de saisie et activer ou désactiver l'interruption pour l'invite initiale et les nouvelles tentatives. Après chaque tentative de capture de l'entrée, Amazon Lex V2 peut appeler la fonction Lambda configurée pour le bot avec une étiquette d'invocation fournie pour les nouvelles tentatives. Vous pouvez utiliser la fonction Lambda, par exemple, pour appliquer votre logique métier afin de tenter de la résoudre en une valeur valide. Cette fonction Lambda peut être activée dansOptions avancéespour les instructions relatives aux créneaux.



Vous pouvez définir les réponses que le bot doit envoyer à l'utilisateur une fois que la valeur du slot est saisie ou si le nombre maximum de tentatives est dépassé. Par exemple, pour un robot chargé de planifier l'entretien d'une voiture, vous pouvez envoyer un message à l'utilisateur lorsque le numéro d'identification du véhicule (VIN) est saisi :

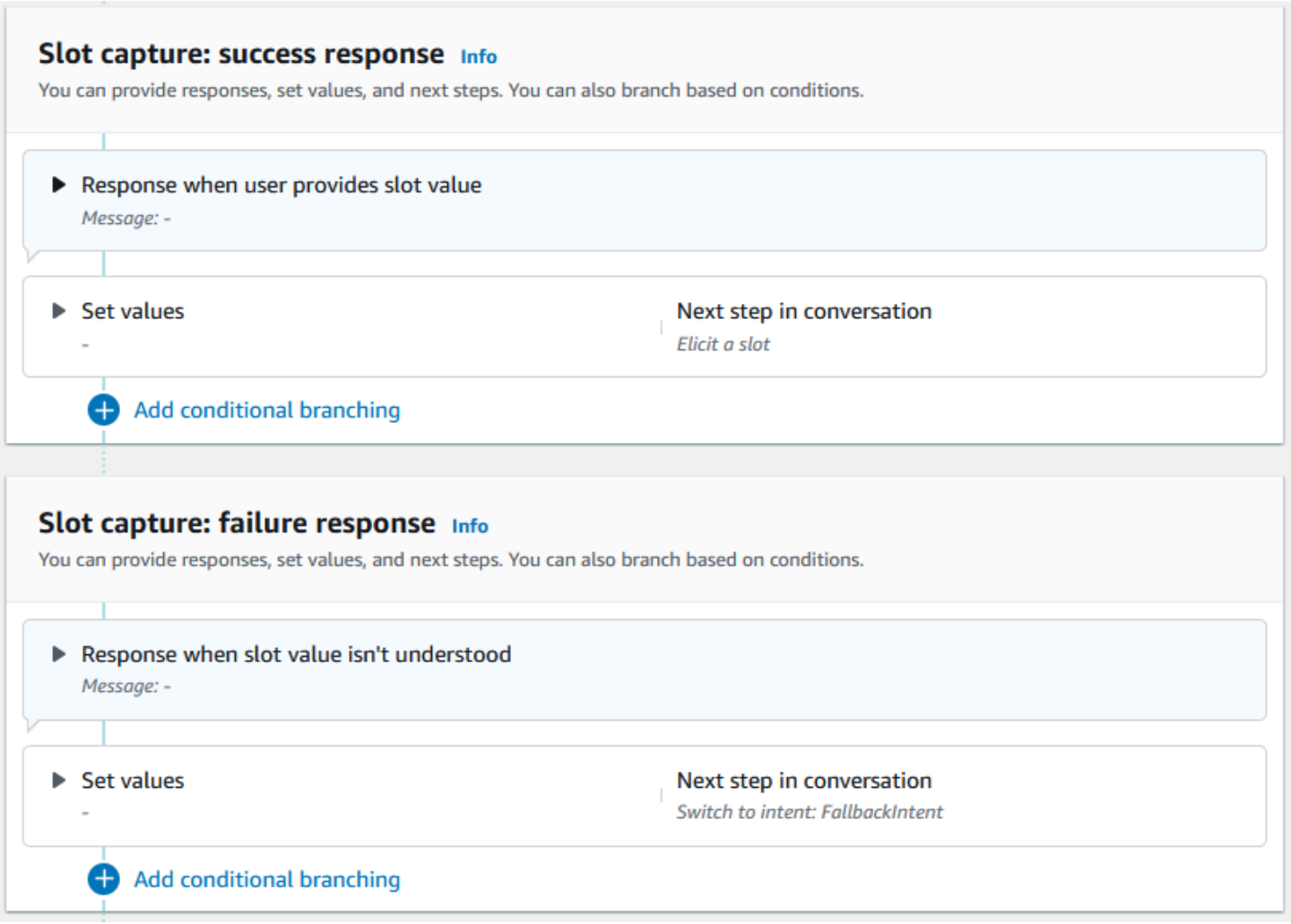
> Merci d'avoir fourni le numéro VIN de votre véhicule. Je vais maintenant fixer un rendez-vous.

Vous pouvez créer deux réponses :

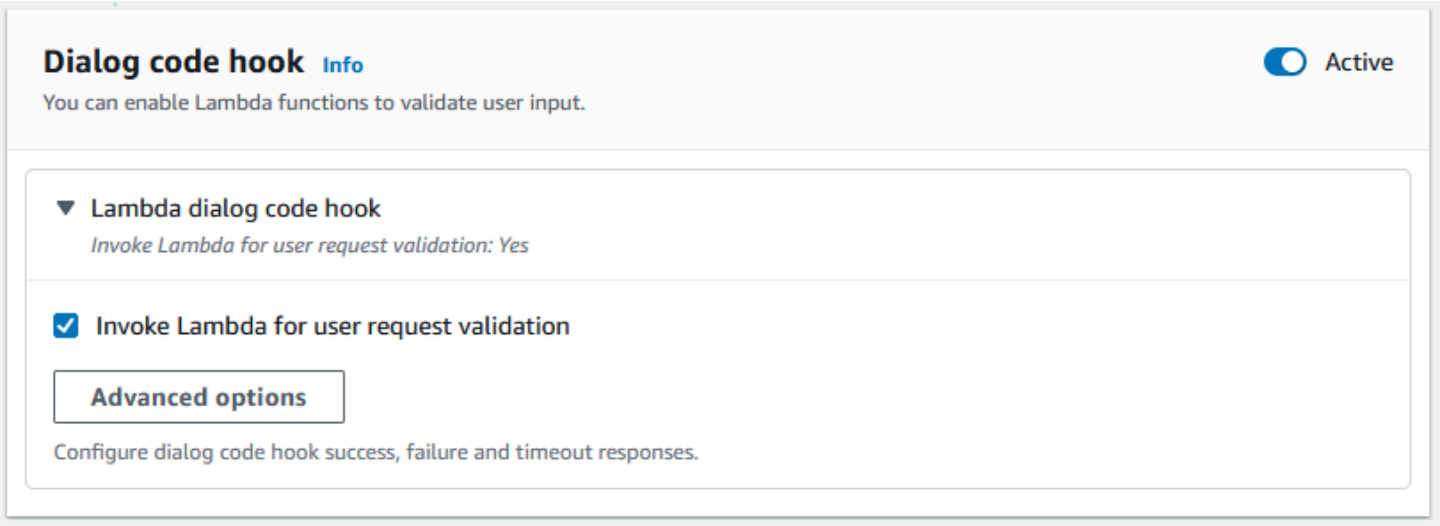
- Réponse positive— envoyé lorsqu'Amazon Lex V2 comprend la valeur d'un emplacement.
- Réponse en cas de panne— envoyé lorsqu'Amazon Lex V2 ne parvient pas à comprendre la valeur d'un emplacement fournie par l'utilisateur après le nombre maximum de tentatives.

Vous pouvez définir des valeurs, configurer les étapes suivantes et appliquer des conditions correspondant à chaque réponse pour concevoir le flux de conversation.

En l'absence d'une condition ou d'une étape suivante explicite, Amazon Lex V2 passe au créneau suivant par ordre de priorité.



Vous pouvez utiliser une fonction Lambda pour valider la valeur d'un slot saisie par un utilisateur et déterminer la prochaine action à effectuer. Par exemple, vous pouvez utiliser la fonction de validation pour vous assurer que la valeur saisie se situe dans la plage correcte ou qu'elle est correctement formatée. Pour activer la fonction Lambda, choisissezInvoquer la fonction Lambdacase à cocher etActifbouton dans leCrochet de code de dialoguesection. Vous pouvez spécifier une étiquette d'invocation pour le crochet du code de boîte de dialogue. Cette étiquette d'invocation peut être utilisée dans la fonction Lambda pour écrire la logique métier correspondant à l'élicitation du créneau.



Les créneaux qui ne sont pas nécessaires pour l'intention ne font pas partie du flux de conversation principal. Toutefois, si un énoncé utilisateur contient une valeur que votre bot identifie comme correspondant à un emplacement facultatif, il peut remplir l'emplacement avec cette valeur. Par exemple, si vous configurez un bot de business intelligence pour qu'il dispose d'une option`City`slot et énoncé de l'utilisateur**`What is the sales for April in San Diego?`**, le bot remplit l'emplacement optionnel avec**`San Diego`**. Vous pouvez configurer la logique métier pour utiliser la valeur d'emplacement facultative, si elle est présente.

Les créneaux non requis pour l'intention ne peuvent pas être obtenus lors des étapes suivantes. Ces étapes peuvent être renseignées uniquement lors de l'élicitation de l'intention (comme dans l'exemple précédent) ou peuvent être déclenchées en définissant l'état de la boîte de dialogue dans la fonction Lambda. Si le créneau est obtenu à l'aide de la fonction Lambda, vous devez utiliser la fonction Lambda pour décider de l'étape suivante de la conversation une fois l'élicitation du créneau terminée. Pour activer la prise en charge de l'étape suivante lors de la création du bot, vous devez marquer l'emplacement correspondant à l'intention.

> **Note**
>
> Le 17 août 2022, Amazon Lex V2 a publié une modification de la façon dont les conversations sont gérées avec l'utilisateur. Cette modification vous permet de mieux contrôler le chemin emprunté par l'utilisateur tout au long de la conversation. Pour plus d'informations, veuillez consulter Comprendre la gestion du flux de conversation. Les robots créés avant le 17 août 2022 ne prennent pas en charge les messages d'accroche à code de dialogue, la définition de valeurs, la configuration des étapes suivantes et l'ajout de conditions.

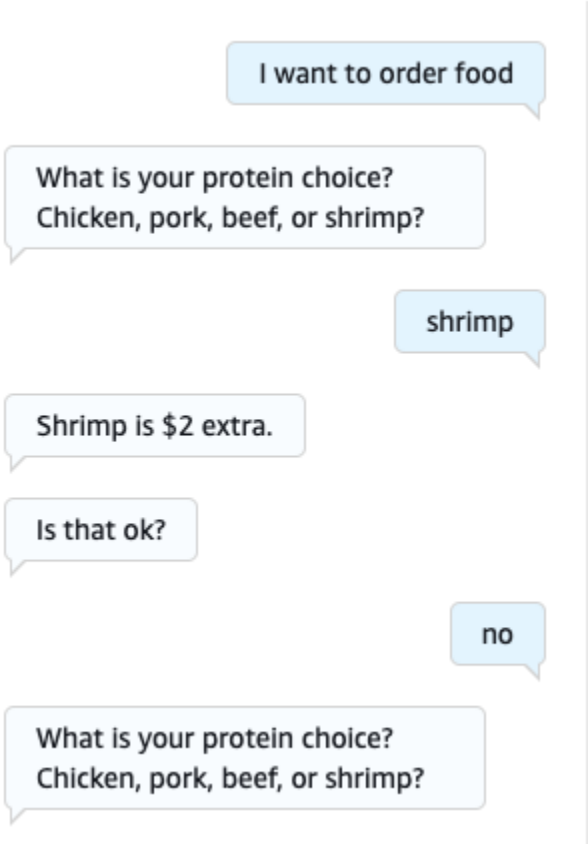
Les rubriques suivantes décrivent comment configurer un bot pour obtenir à nouveau une valeur d'emplacement qui a déjà été remplie et comment créer un emplacement composé de plusieurs valeurs :

Rubriques

- Des machines à sous stimulantes
- Utilisation de plusieurs valeurs dans un emplacement

Des machines à sous stimulantes

Vous pouvez configurer votre bot pour qu'il sollicite à nouveau un créneau qui a déjà été rempli en définissant la valeur de cet emplacement **`null`** et en configurant l'étape suivante de la conversation pour revenir à l'obtention de ce créneau. Par exemple, vous souhaiterez peut-être demander à nouveau un créneau lorsque votre client a refusé la confirmation de l'offre de créneau sur la base d'informations supplémentaires, comme dans la conversation suivante :



Vous pouvez configurer une boucle à partir de la réponse de confirmation pour réactiver le slot avec l'éditeur d'intention ou le. Utilisation du générateur de conversation visuel

> **Note**
>
> Vous pouvez revenir en arrière pour obtenir à nouveau un créneau à n'importe quel moment de la conversation, à condition de définir au préalable la valeur de ce créneau. **`null`**

Reproduction de l'exemple ci-dessus avec l'éditeur d'intention

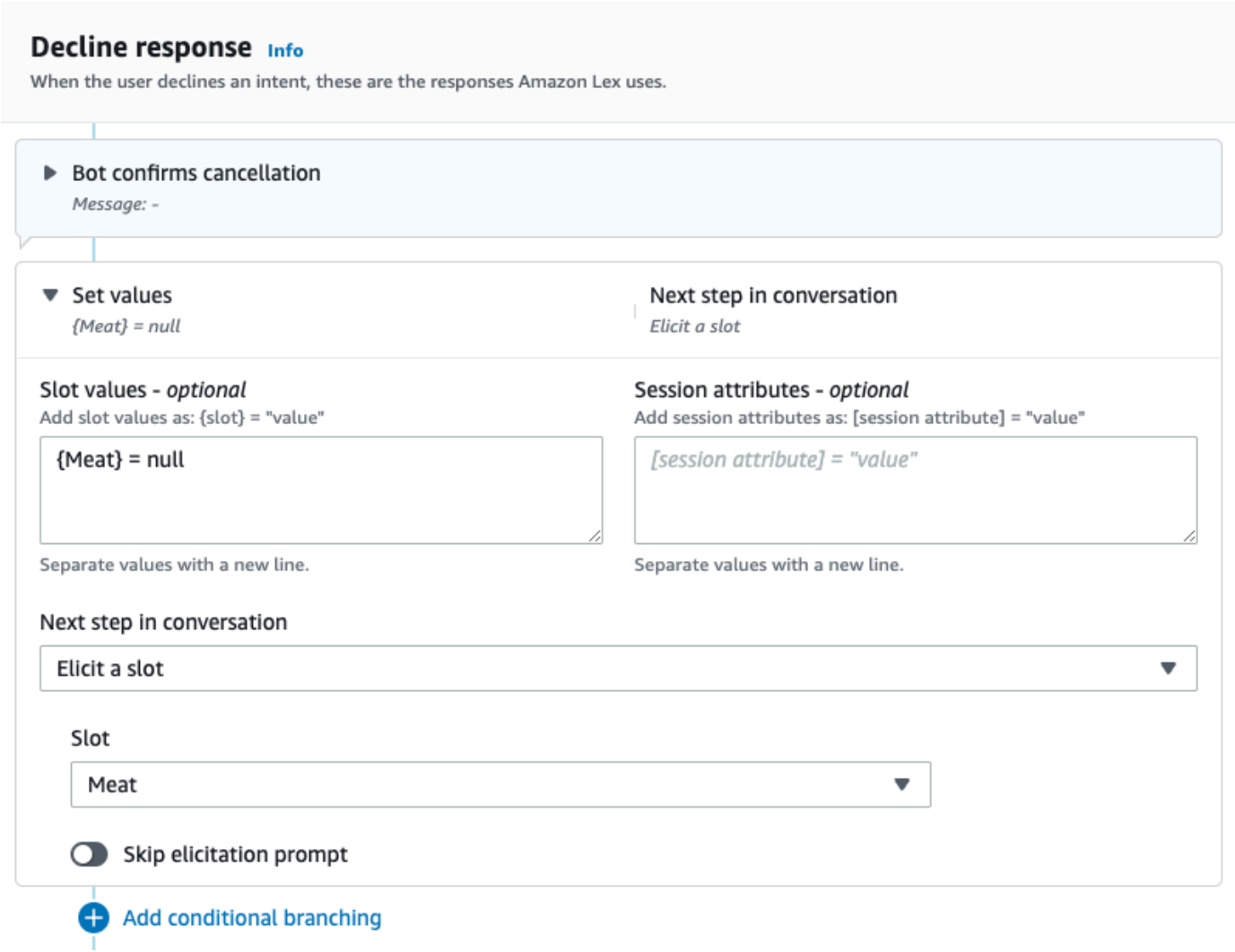
1. Dans la section Confirmation de l'éditeur d'intentions, sélectionnez la flèche droite à côté de Proptes pour confirmer l'intention de développer la section.
2. Sélectionnez Options avancées en bas de la page.
3. Dans la section Refuser la réponse, sélectionnez la flèche droite à côté de Définir les valeurs pour développer la section. Remplissez cette section en suivant les étapes suivantes, comme dans l'image ci-dessous :

    a. Définissez la valeur de l'emplacement que vous souhaitez obtenir à nouveau. **`null`** Dans cet exemple, nous voulons réactiver l'emplacement. Nous saisissons donc les valeurs de l'`Meat`emplacement **`{Meat} = null`** dans la section Valeurs de l'emplacement.

    b. Dans le menu déroulant situé sous Étape suivante de la conversation, choisissez Obtenir un créneau.

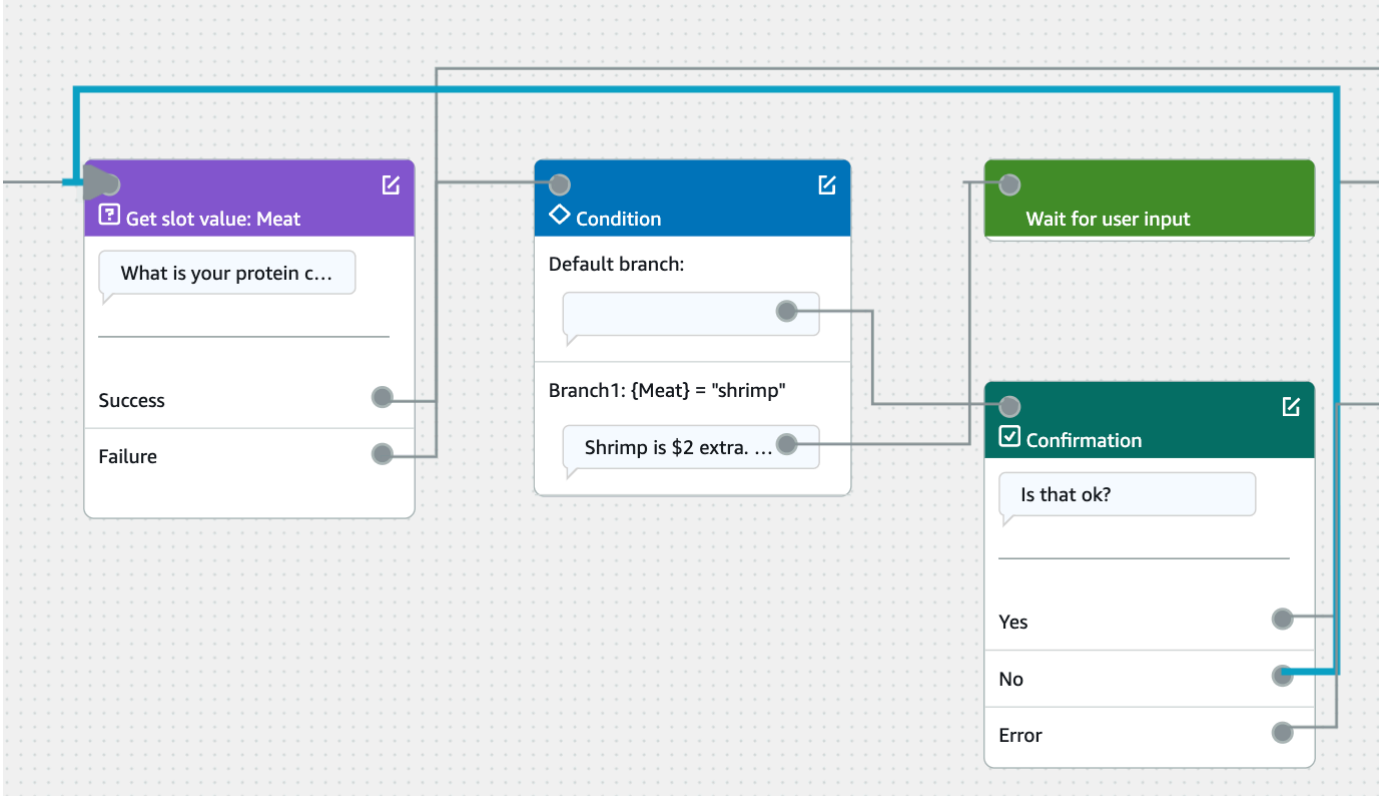
    c. Une section Slot apparaîtra. Dans le menu déroulant situé en dessous, choisissez l'emplacement que vous souhaitez obtenir à nouveau.

    d. Sélectionnez les options de mise à jour pour confirmer vos modifications.

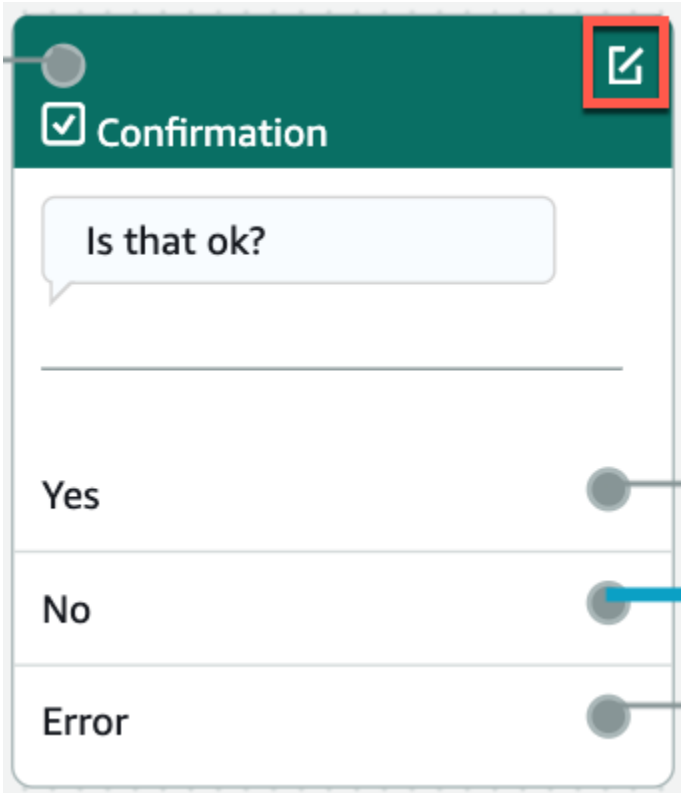


Reproduction de l'exemple ci-dessus avec le générateur de conversation visuel

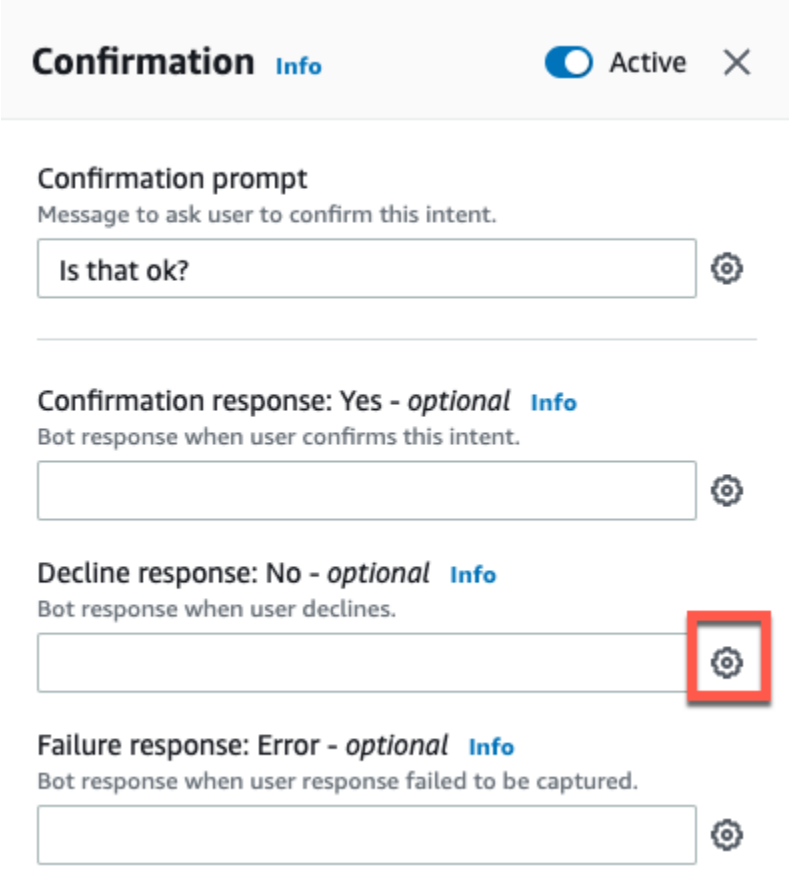
1. Créez une connexion entre le port Non du bloc Confirmation et le port entrant du bloc Get slot value : Meat.



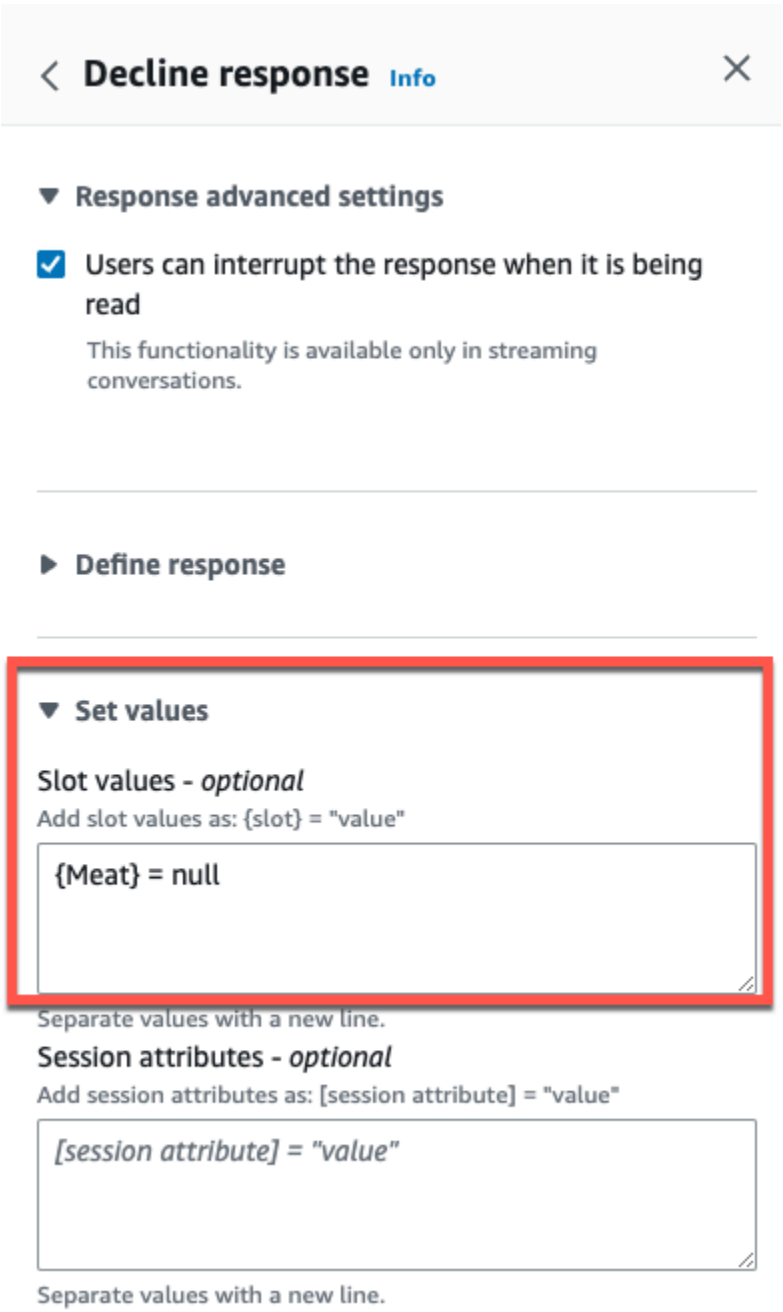
2. Sélectionnez l'icône Modifier dans le coin supérieur droit du bloc de confirmation.



3. Sélectionnez l'icône en forme d'engrenage à côté de la réponse du bot dans la section Refuser la réponse.



4. Dans la section Définir les valeurs, ajoutez « {Meat} = null » dans la zone Valeurs du slot.



5. Sélectionnez Enregistrer l'intention.

Utilisation de plusieurs valeurs dans un emplacement

> **Note**
>
> Les emplacements à valeurs multiples ne sont pris en charge qu'en anglais (États-Unis).

Dans certains cas, vous souhaiterez peut-être capturer plusieurs valeurs pour un seul emplacement. Par exemple, un robot de commande de pizzas peut avoir une intention avec l'énoncé suivant :

```
I want a pizza with {toppings}
```

L'intention est que l'`{toppings}`emplacement contienne une liste des garnitures que le client souhaite ajouter à sa pizza, par exemple du « pepperoni et de l'ananas ».

Pour configurer un slot afin de capturer plusieurs valeurs, vous devez définir le `allowMultipleValues` champ du slot sur true. Vous pouvez définir le champ à l'aide de la console ou à l'aide de l'UpdateSlotopération CreateSlotor.

Vous pouvez uniquement marquer des emplacements avec des types d'emplacements personnalisés comme des emplacements à valeurs multiples.

Pour un emplacement à valeurs multiples, Amazon Lex V2 renvoie une liste de valeurs d'emplacement en réponse à l'RecognizeUtteranceopération RecognizeTextor. Voici les informations d'emplacement renvoyées par le bot pour l'énoncé « Je veux une pizza au pepperoni et à l'ananas ». OrderPizza

```
    "slots": {
        "toppings": {
            "shape": "List",
            "value": {
                "interpretedValue": "pepperoni and pineapple",
                "originalValue": "pepperoni and pineapple",
                "resolvedValues": [
                    "pepperoni and pineapple"
                ]
            },
            "values": [
                {
                    "shape": "Scalar",
                    "value": {
                        "interpretedValue": "pepperoni",
                        "originalValue": "pepperoni",
                        "resolvedValues": [
                            "pepperoni"
                        ]
                    }
                },
```

```
                {
                    "shape": "Scalar",
                    "value:": {
                        "interpretedValue": "pineapple",
                        "originalValue": "pineapple",
                        "resolvedValues": [
                            "pineapple"
                        ]
                    }
                }
            ]
        }
    }
```

Les emplacements à valeurs multiples renvoient toujours une liste de valeurs. Lorsque l'énoncé ne contient qu'une seule valeur, la liste des valeurs renvoyées ne contient qu'une seule réponse.

Amazon Lex V2 reconnaît plusieurs valeurs séparées par des espaces, des virgules (,) et la conjonction « et ». Les emplacements à valeurs multiples fonctionnent à la fois avec la saisie de texte et la saisie vocale.

Vous pouvez utiliser des emplacements à valeurs multiples dans les invites. Par exemple, vous pouvez configurer l'invite de confirmation pour une intention de

```
Would you like me to order your {toppings} pizza?
```

Lorsqu'Amazon Lex V2 envoie le message à l'utilisateur, celui-ci envoie le message suivant : « Aimeriez-vous que je commande votre pizza au pepperoni et à l'ananas ? »

Les slots à valeurs multiples prennent en charge des valeurs par défaut uniques. Si plusieurs valeurs par défaut sont fournies, Amazon Lex V2 renseigne l'emplacement avec uniquement la première valeur disponible. Pour plus d'informations, veuillez consulter Utilisation des valeurs d'emplacement par défaut.

Vous pouvez utiliser l'obfuscation des emplacements pour masquer les valeurs d'un emplacement à valeurs multiples dans les journaux de conversation. Lorsque vous masquez des valeurs d'emplacement, la valeur de chacune des valeurs d'emplacement est remplacée par le nom de l'emplacement. Pour plus d'informations, veuillez consulter Masquer les valeurs des créneaux dans les journaux de conversation.

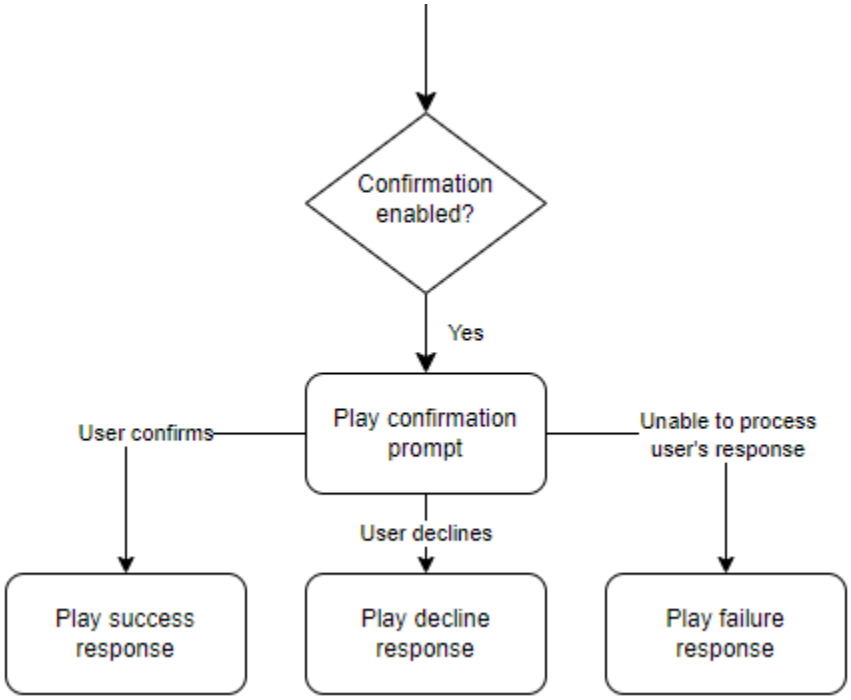
Confirmation

Une fois que la conversation avec l'utilisateur est terminée et que les valeurs des créneaux correspondant à l'intention sont remplies, vous pouvez configurer une invite de confirmation pour demander à l'utilisateur si les valeurs des créneaux sont correctes. Par exemple, un robot qui planifie des rendez-vous d'entretien pour les voitures peut demander à l'utilisateur ce qui suit :

> L'entretien de votre Honda Civic 2017 est prévu pour le 25 mars à 15 h. Est-ce que c'est bon ?

Vous pouvez définir 3 types de réponses à l'invite de confirmation :

- Réponse de confirmation— Cette réponse est envoyée à l'utilisateur lorsque celui-ci confirme son intention. Par exemple, une fois que l'utilisateur a répondu « oui » à l'invite « Voulez-vous passer la commande ? »
- Réponse de refus— Cette réponse est envoyée à l'utilisateur lorsque celui-ci refuse son intention. Par exemple, une fois que l'utilisateur a répondu « non » à la question « Voulez-vous passer la commande ? »
- Réponse en cas de panne— Cette réponse est envoyée à l'utilisateur lorsque l'invite de confirmation ne peut pas être traitée. Par exemple, si la réponse de l'utilisateur n'a pas pu être comprise ou si elle n'a pas pu être résolue par un oui ou un non.

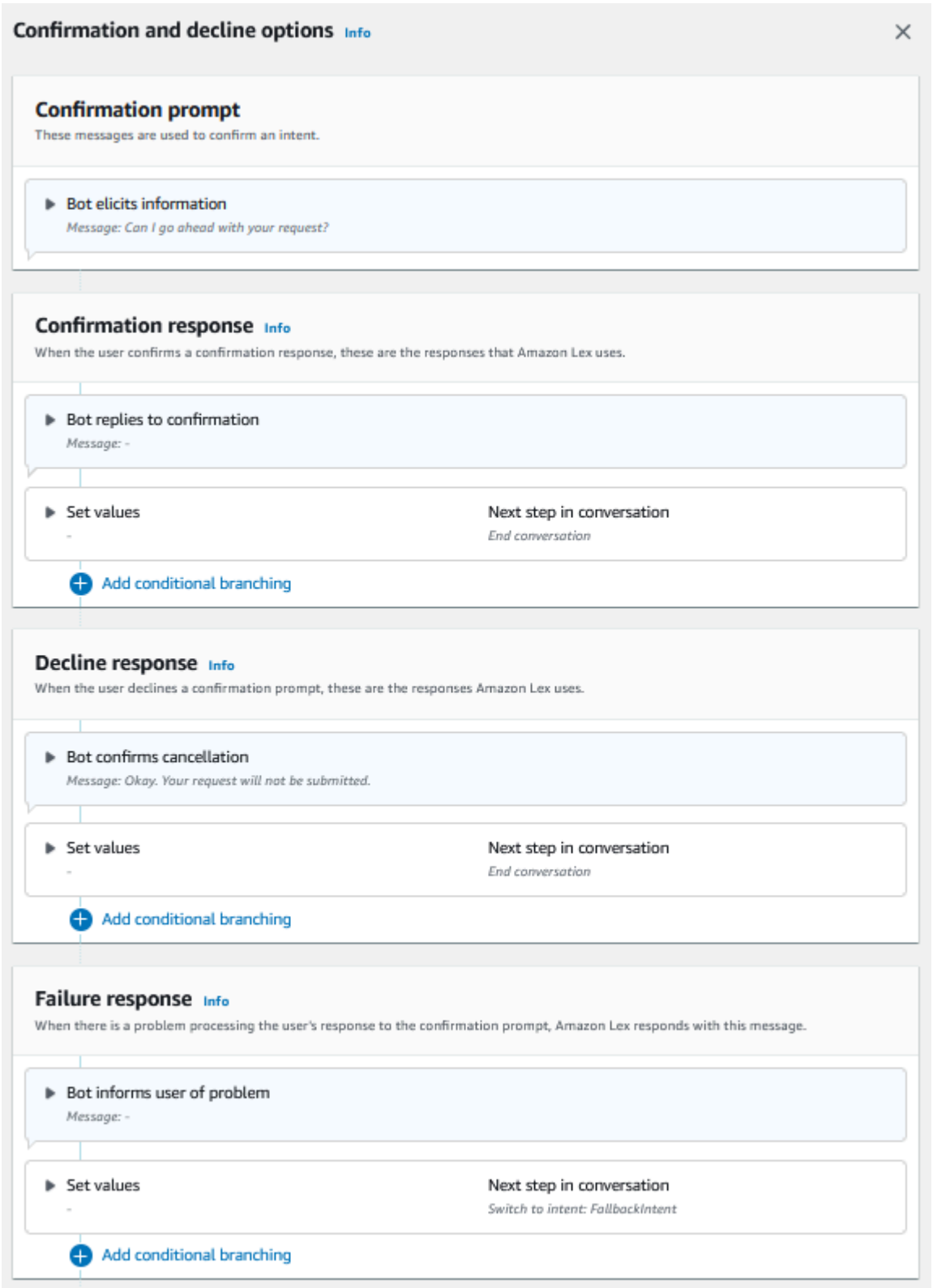


Si vous ne spécifiez pas d'invite de confirmation, Amazon Lex V2 passe à l'étape de traitement ou à la réponse de clôture.

Vous pouvez définir des valeurs, configurer les étapes suivantes et appliquer des conditions correspondant à chaque réponse pour concevoir le flux de conversation. En l'absence d'une condition ou d'une étape suivante explicite, Amazon Lex V2 passe à l'étape d'expédition.

Vous pouvez également activer le crochet à code de dialogue pour valider les informations saisies dans l'intention avant de l'envoyer pour exécution. Pour utiliser un crochet de code, activez le crochet de code de la boîte de dialogue dans les options avancées de l'invite de confirmation. Configurez également l'étape suivante de l'état précédent pour exécuter le hook du code de dialogue. Pour plus d'informations, veuillez consulter Invoquer le crochet de code de dialogue.

> **Note**
>
> Si vous utilisez un crochet de code pour déclencher l'étape de confirmation lors de l'exécution, vous devez marquer l'étape de confirmation commeActifau moment de la construction.


Confirmation and decline options Info
Confirmation prompt
These messages are used to confirm an intent.
Bot elicits information
Message: Can I go ahead with your request?
Confirmation response Info
When the user confirms a confirmation response, these are the responses that Amazon Lex uses.
Bot replies to confirmation
Message: -
Set values
-
Next step in conversation
End conversation
Add conditional branching
Decline response Info
When the user declines a confirmation prompt, these are the responses Amazon Lex uses.
Bot confirms cancellation
Message: Okay. Your request will not be submitted.
Set values
-
Next step in conversation
End conversation
Add conditional branching
Failure response Info
When there is a problem processing the user's response to the confirmation prompt, Amazon Lex responds with this message.
Bot informs user of problem
Message: -
Set values
-
Next step in conversation
Switch to intent: FallbackIntent
Add conditional branching

> **Note**
>
> Le 17 août 2022, Amazon Lex V2 a publié une modification de la façon dont les conversations sont gérées avec l'utilisateur. Cette modification vous permet de mieux contrôler le chemin emprunté par l'utilisateur tout au long de la conversation. Pour plus d'informations, veuillez consulter Comprendre la gestion du flux de conversation. Les robots créés avant le 17 août 2022 ne prennent pas en charge les messages d'accroche à code de dialogue, la définition de valeurs, la configuration des étapes suivantes et l'ajout de conditions.

Utilisation d'une fonction Lambda pour valider une intention.

Vous pouvez définir un hook de code Lambda pour valider l'intention avant de l'envoyer pour exécution. Pour utiliser un crochet de code, activez le crochet de code de la boîte de dialogue dans les options avancées de l'invite de confirmation.

Lorsque vous utilisez un crochet de code, vous pouvez définir les actions qu'Amazon Lex V2 entreprend après l'exécution du crochet de code. Vous pouvez créer trois types de réponses :

- Réponse positive— Envoyé à l'utilisateur lorsque le code hook se termine correctement.
- Réponse en cas de panne— Envoyé à l'utilisateur lorsque le code hook ne s'exécute pas correctement ou lorsque le code hook revient`Failure`dans la réponse.
- Délai de réponse— Envoyé à l'utilisateur lorsque le code hook ne se termine pas dans le délai d'expiration configuré.

Exécution

Une fois que toutes les valeurs d'emplacement ont été fournies par l'utilisateur conformément à l'intention, Amazon Lex V2 répond à la demande de l'utilisateur. Vous pouvez configurer les options suivantes pour le traitement des commandes.

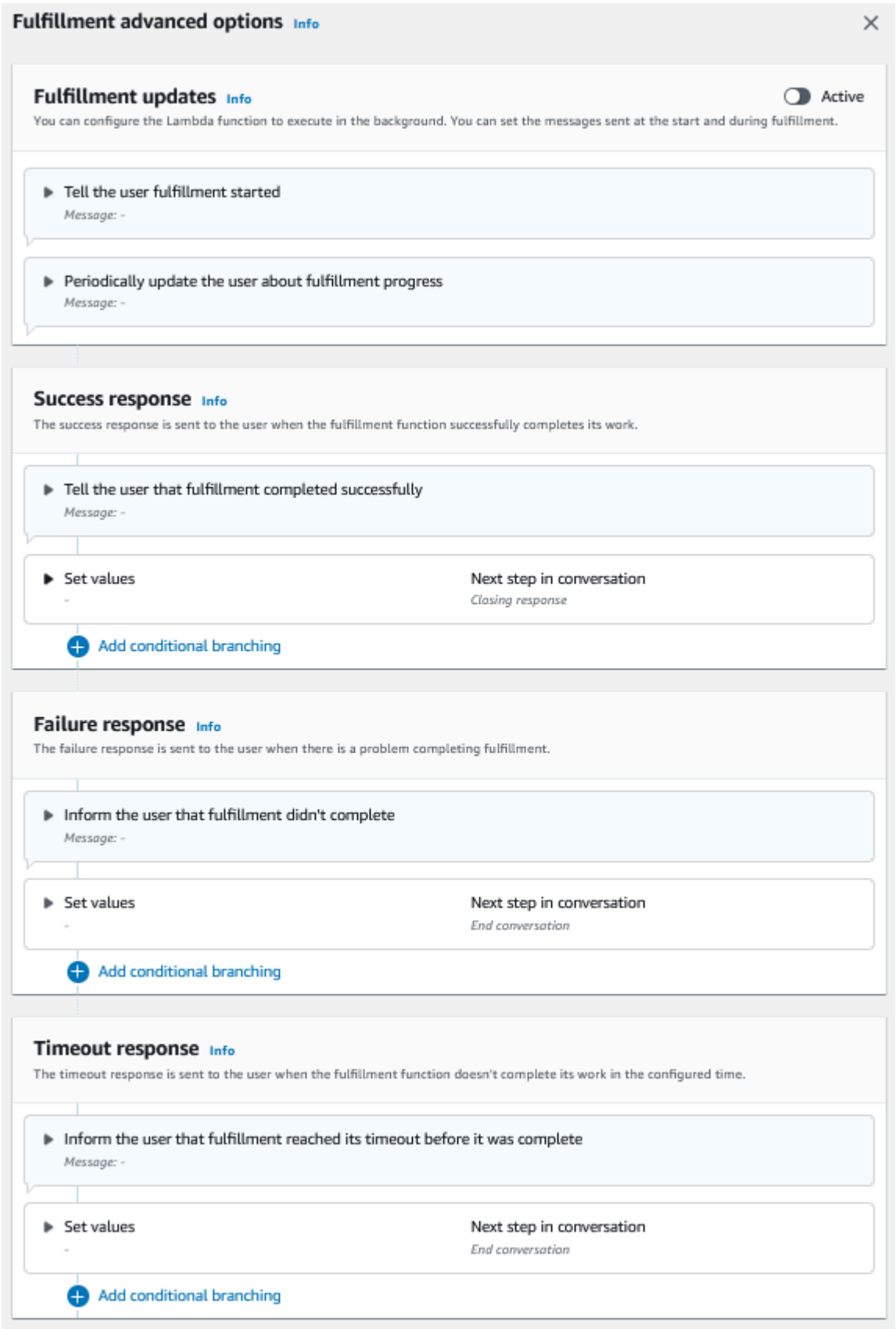
- Crochet à code d'expédition— Vous pouvez utiliser cette option pour contrôler l'appel Lambda d'exécution. Si l'option est désactivée, l'exécution aboutit sans invoquer la fonction Lambda.
- Mises à jour sur— Vous pouvez activer les mises à jour d'exécution pour les fonctions Lambda dont l'exécution prend plus de quelques secondes, afin que l'utilisateur sache que le processus est en cours. Pour plus d'informations, veuillez consulter Configuration des mises à jour de progression du traitement. Cette fonctionnalité n'est disponible que pour les conversations en streaming.

- Réponses relatives à l'exécution— Vous pouvez configurer une réponse de réussite, une réponse d'échec et une réponse de délai d'expiration. La réponse appropriée est renvoyée à l'utilisateur en fonction de l'état de l'appel Lambda d'exécution.

Il existe trois réponses possibles à l'exécution des commandes :

- Réponse positive— Un message envoyé lorsque le traitement Lambda s'est terminé avec succès.
- Réponse en cas de panne— Un message envoyé si le traitement a échoué ou si Lambda ne peut pas être terminé pour une raison quelconque.
- Délai de réponse— Un message envoyé si la fonction Lambda d'exécution ne se termine pas dans le délai configuré.

Vous pouvez définir des valeurs, configurer les étapes suivantes et appliquer des conditions correspondant à chaque réponse pour concevoir le flux de conversation. En l'absence d'une condition ou d'une étape suivante explicite, Amazon Lex V2 passe à la réponse finale.


Fulfillment advanced options Info
Fulfillment updates Info
Active
You can configure the Lambda function to execute in the background. You can set the messages sent at the start and during fulfillment.
Tell the user fulfillment started
Message: -
Periodically update the user about fulfillment progress
Message: -
Success response Info
The success response is sent to the user when the fulfillment function successfully completes its work.
Tell the user that fulfillment completed successfully
Message: -
Set values
-
Next step in conversation
Closing response
Add conditional branching
Failure response Info
The failure response is sent to the user when there is a problem completing fulfillment.
Inform the user that fulfillment didn't complete
Message: -
Set values
-
Next step in conversation
End conversation
Add conditional branching
Timeout response Info
The timeout response is sent to the user when the fulfillment function doesn't complete its work in the configured time.
Inform the user that fulfillment reached its timeout before it was complete
Message: -
Set values
-
Next step in conversation
End conversation
Add conditional branching

> **Note**
>
> Le 17 août 2022, Amazon Lex V2 a publié une modification de la façon dont les conversations sont gérées avec l'utilisateur. Cette modification vous permet de mieux contrôler le chemin emprunté par l'utilisateur tout au long de la conversation. Pour plus d'informations, veuillez consulter Comprendre la gestion du flux de conversation. Les robots créés avant le 17 août 2022 ne prennent pas en charge les messages d'accroche à code de dialogue, la définition de valeurs, la configuration des étapes suivantes et l'ajout de conditions.

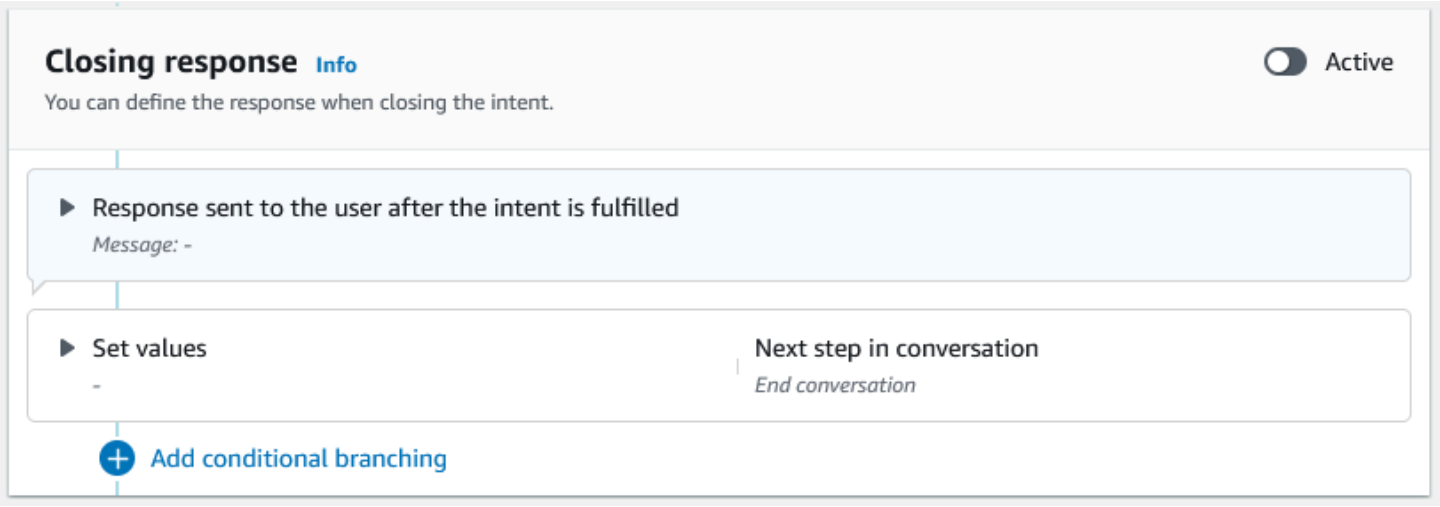
Réponse finale

La réponse finale est envoyée à votre utilisateur une fois que son intention est satisfaite. Vous pouvez utiliser la réponse finale pour mettre fin à la conversation, ou vous pouvez l'utiliser pour faire savoir à l'utilisateur qu'il peut poursuivre avec une autre intention. Par exemple, dans un robot de réservation de voyages, vous pouvez définir la réponse finale pour l'intention de réserver une chambre d'hôtel comme suit :

```
Très bien, j'ai réservé ta chambre d'hôtel. Y a-t-il autre chose que je puisse faire pour vous aider ?
```

Vous pouvez définir des valeurs, configurer les étapes suivantes et appliquer des conditions après la réponse finale pour concevoir le chemin de conversation. En l'absence d'une condition ou d'une étape suivante explicite, Amazon Lex V2 met fin à la conversation.

Si vous ne fournissez pas de réponse finale ou si aucune des conditions n'est jugée vraie, Amazon Lex V2 met fin à la conversation avec votre bot.

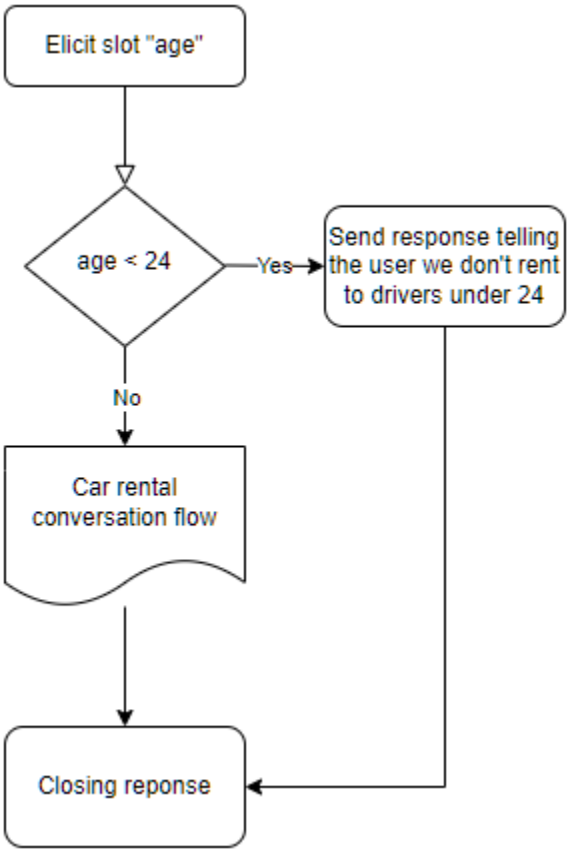


> **Note**
>
> Le 17 août 2022, Amazon Lex V2 a publié une modification de la façon dont les conversations sont gérées avec l'utilisateur. Cette modification vous permet de mieux contrôler le chemin emprunté par l'utilisateur tout au long de la conversation. Pour plus d'informations, veuillez consulter Comprendre la gestion du flux de conversation. Les robots créés avant le 17 août 2022 ne prennent pas en charge les messages d'accroche à code de dialogue, la définition de valeurs, la configuration des étapes suivantes et l'ajout de conditions.

Création de parcours de conversation

Amazon Lex V2 gère généralement le flux des conversations avec vos utilisateurs. Pour les robots simples, le flux par défaut peut suffire à créer une bonne expérience pour vos utilisateurs. Toutefois, pour les robots plus complexes, vous souhaiterez peut-être prendre le contrôle de la conversation et orienter le flux vers des chemins plus complexes.

Par exemple, dans un bot qui réserve des locations de voitures, il est possible que vous ne louiez pas à de jeunes conducteurs. Dans ce cas, vous pouvez créer une condition qui vérifie si un conducteur n'a pas atteint un certain âge et, dans l'affirmative, passer à la réponse finale.



Pour concevoir de telles interactions, vous pouvez configurer l'étape suivante à chaque étape de la conversation, évaluer les conditions, définir des valeurs et invoquer des crochets de code.

Le branchement conditionnel vous aide à créer des chemins pour vos utilisateurs par le biais d'interactions complexes. Vous pouvez utiliser une branche conditionnelle à tout moment lorsque vous passez le contrôle de la conversation à votre bot. Par exemple, vous pouvez créer une condition avant que le bot n'obtienne la première valeur d'intervalle, vous pouvez créer une condition entre chaque valeur d'intervalle ou vous pouvez créer une condition avant que le bot ne ferme la conversation. Pour obtenir la liste des lieux auxquels vous pouvez ajouter des conditions, consultezAjouter des intentions.

Lorsque vous créez un bot, Amazon Lex V2 crée un chemin par défaut dans la conversation en fonction de l'ordre de priorité des emplacements. Pour personnaliser le parcours de conversation, vous pouvez modifier l'étape suivante à tout moment de la conversation. Pour plus d’informations, consultez Configurer les prochaines étapes de la conversation.

Pour créer des chemins alternatifs en fonction des conditions, vous pouvez utiliser une branche conditionnelle à tout moment de la conversation. Par exemple, vous pouvez créer une condition avant que le bot n'obtienne la première valeur d'emplacement. Vous pouvez créer une condition entre

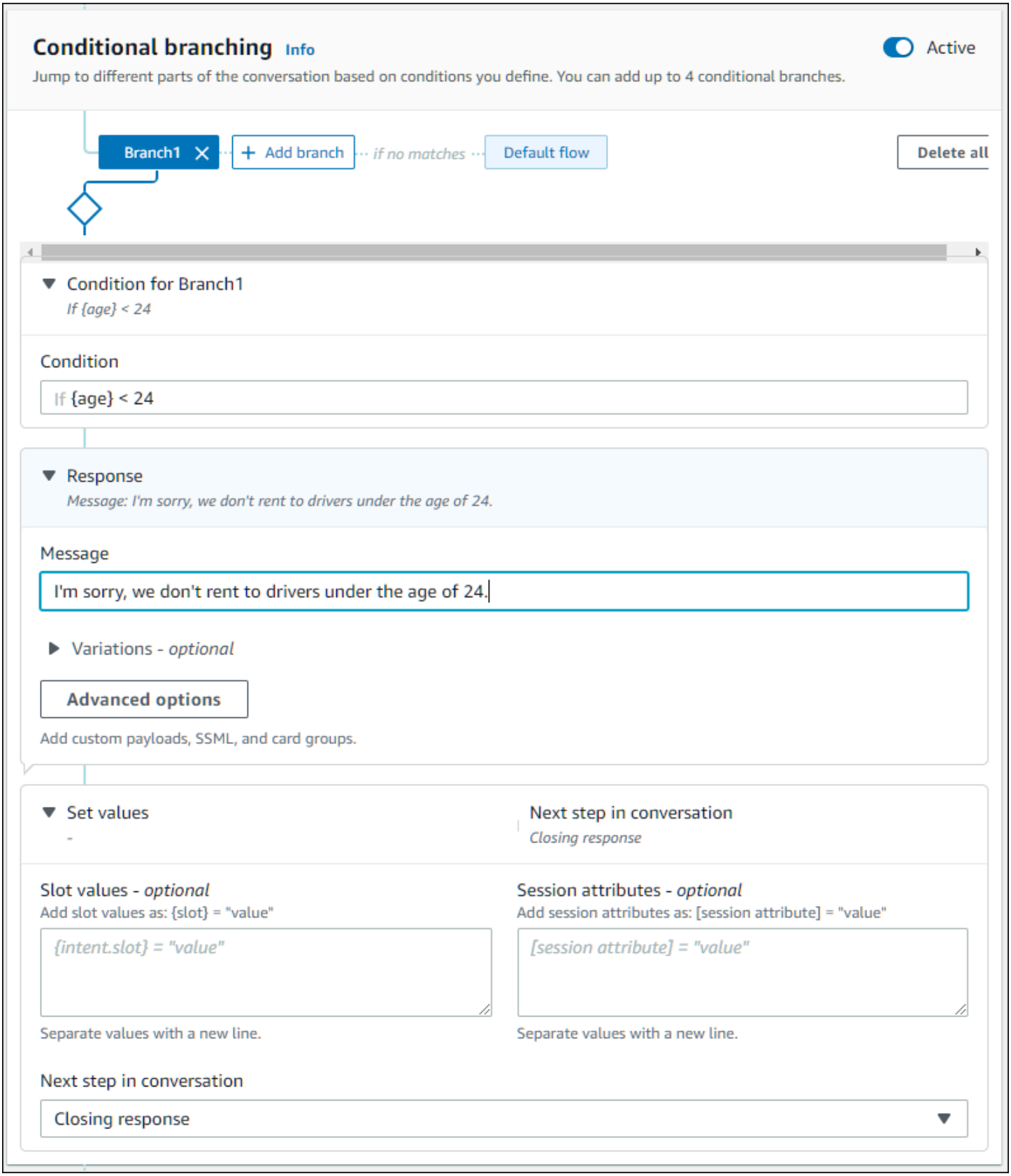
l'obtention de la valeur de chaque emplacement, ou vous pouvez créer une condition avant que le bot ne ferme la conversation. Pour consulter la liste des lieux où vous pouvez ajouter des conditions, consultezAjouter des conditions aux conversations dans les succursales.

Vous pouvez définir des conditions en fonction des valeurs des créneaux, des attributs de session, du mode de saisie et de la transcription d'entrée, ou d'une réponse d'Amazon Kendra.

Vous pouvez définir les valeurs des attributs d'emplacement et de session à chaque étape de la conversation. Pour plus d'informations, consultez Définissez des valeurs au cours de la conversation.

Vous pouvez également définir l'action suivante sur Dialog Code Hook pour exécuter une fonction Lambda. Pour plus d'informations, consultez Invoquer le crochet de code de dialogue.

L'image suivante montre la création d'un chemin pour un emplacement dans la console. Dans cet exemple, Amazon Lex V2 indiquera l' « âge » de l'emplacement. Si la valeur de l'emplacement est inférieure à 24, Amazon Lex V2 passe à la réponse de fermeture, sinon Amazon Lex suivra le chemin par défaut.


Conditional branching Info
Active
Jump to different parts of the conversation based on conditions you define. You can add up to 4 conditional branches.
Branch1
Add branch
if no matches
Default flow
Delete all
Condition for Branch1
If {age} < 24
Condition
If {age} < 24
Response
Message: I'm sorry, we don't rent to drivers under the age of 24.
Message
I'm sorry, we don't rent to drivers under the age of 24.
Variations - optional
Advanced options
Add custom payloads, SSML, and card groups.
Set values
-
Next step in conversation
Closing response
Slot values - optional
Add slot values as: {slot} = "value"
{intent.slot} = "value"
Separate values with a new line.
Session attributes - optional
Add session attributes as: [session attribute] = "value"
[session attribute] = "value"
Separate values with a new line.
Next step in conversation
Closing response

> **Note**
>
> Le 17 août 2022, Amazon Lex V2 a publié une modification de la façon dont les conversations sont gérées avec l'utilisateur. Cette modification vous permet de mieux contrôler le chemin emprunté par l'utilisateur tout au long de la conversation. Pour plus d'informations, consultez Comprendre la gestion du flux de conversation. Les robots créés avant le 17 août 2022 ne prennent pas en charge les messages d'accrochage par code de dialogue, la définition de valeurs, la configuration des étapes suivantes et l'ajout de conditions.

Configurer les prochaines étapes de la conversation

Vous pouvez configurer une étape suivante à chaque étape de la conversation pour concevoir les conversations. En général, Amazon Lex V2 configure automatiquement les prochaines étapes par défaut pour chaque étape de la conversation selon l'ordre suivant.

Réponse initiale → Sollicitation du créneau → Confirmation (si active) → Exécution (si active) → Réponse de clôture (si active) → Fin de la conversation

Vous pouvez modifier les prochaines étapes par défaut et concevoir la conversation en fonction de l'expérience utilisateur attendue. Les étapes suivantes peuvent être configurées à chaque étape de la conversation :

Sauter à

- Réponse initiale — La conversation est relancée depuis le début de l'intention. Vous pouvez choisir d'ignorer la réponse initiale lors de la configuration de l'étape suivante
- Obtenir un emplacement — Vous pouvez obtenir n'importe quel emplacement dans l'intention.
- Évaluer les conditions — Vous pouvez évaluer les conditions et partager la conversation à n'importe quelle étape de la conversation.
- Invoquer le code hook du dialogue : vous pouvez invoquer la logique métier à n'importe quelle étape.
- Confirmer l'intention — L'utilisateur sera invité à confirmer l'intention.
- Réalisation de l'intention — La réalisation de l'intention débutera à l'étape suivante.
- Réponse de clôture — La réponse de clôture sera renvoyée à l'utilisateur.

Basculer vers

- Intention — Vous pouvez passer à une intention différente et poursuivre la conversation sur cette intention. Vous pouvez éventuellement ignorer la réponse initiale à l'intention lors de la transition.
- Intention : emplacement spécifique — Vous pouvez directement obtenir un emplacement spécifique dans une intention différente si vous avez déjà capturé certaines valeurs d'emplacement dans l'intention actuelle.

Attendre les entrées de l'utilisateur : le bot attend que l'utilisateur fournisse des entrées pour reconnaître toute nouvelle intention. Vous pouvez configurer des messages tels que « Y a-t-il autre chose que je puisse faire pour vous aider ? » avant de définir cette étape suivante. Le bot sera en état `ElicitIntent` de dialogue.

Fin de la conversation — La conversation avec le bot est close.

> **Note**
>
> Le 17 août 2022, Amazon Lex V2 a publié une modification de la façon dont les conversations sont gérées avec l'utilisateur. Cette modification vous permet de mieux contrôler le chemin emprunté par l'utilisateur tout au long de la conversation. Pour plus d'informations, consultez Comprendre la gestion du flux de conversation. Les robots créés avant le 17 août 2022 ne prennent pas en charge les messages d'accrochage par code de dialogue, la définition de valeurs, la configuration des étapes suivantes et l'ajout de conditions.

Définissez des valeurs au cours de la conversation

Amazon Lex V2 permet de définir des valeurs d'emplacement et des valeurs d'attributs de session à chaque étape de la conversation. Vous pouvez ensuite utiliser ces valeurs au cours de la conversation pour évaluer les conditions ou les utiliser lors de la réalisation d'une intention.

Vous pouvez définir des valeurs de créneau pour l'intention actuelle. Si l'étape suivante de la conversation consiste à invoquer une autre intention, vous pouvez définir des valeurs d'intervalle pour cette nouvelle intention.

Si le slot attribué n'est pas rempli ou si le chemin JSON ne peut pas être analysé, l'attribut sera défini sur. `null`

Utilisez la syntaxe suivante lorsque vous utilisez des valeurs d'emplacement et des attributs de session :

- Valeurs des emplacements : entourez le nom des emplacements entre accolades (« {} »). Pour les valeurs d'emplacement dans l'intention actuelle, il suffit d'utiliser le nom de l'emplacement. Par exemple, `{slot}`. Si vous définissez une valeur dans l'intention suivante, vous devez utiliser à la fois le nom de l'intention et le nom du slot pour identifier le slot. Par exemple, `{intent.slot}`.

  Exemples :

  - `{PhoneNumber} = "1234567890"`
  - `{CheckBalance.AccountNumber} = "99999999"`
  - `{BookingID} = "ABC123"`
  - `{FirstName} = "John"`

  La valeur d'un emplacement peut être l'une des suivantes :

  - une chaîne constante
  - un chemin JSON qui fait référence au bloc de transcriptions dans la réponse Amazon Lex (pour en-US et en-GB)
  - un attribut de session

  Exemples :

  - `{username} = "john.doe"`
  - `{username_confidence} = $.transcriptions[0].transcriptionConfidence`
  - `{username_slot_value} = [username]`

  > **Note**
  >
  > Les valeurs des emplacements peuvent également être définies sur`null`. Si vous devez obtenir à nouveau la valeur d'un créneau qui a été rempli, vous devez définir la valeur sur `null` avant de demander à nouveau au client la valeur du créneau. Si le slot attribué n'est pas rempli ou si le chemin JSON ne peut pas être analysé, l'attribut sera défini sur. `null`

- Attributs de session : placez le nom de l'attribut entre crochets (« [] »). Par exemple, `[sessionAttribute]`.

  Exemples :

- `[username] = "john.doe"`
- `[username_confidence] = $.transcriptions[0].transcriptionConfidence`
- `[username_slot_value] = {username}`

La valeur de l'attribut de session peut être l'une des suivantes :

- une chaîne constante
- un chemin JSON qui fait référence au bloc de transcriptions dans la réponse Amazon Lex (pour en-US et en-GB)
- une référence de valeur de slot

> **Note**
>
> Si le slot attribué n'est pas rempli ou si le chemin JSON ne peut pas être analysé, l'attribut sera défini sur. `null`

> **Note**
>
> Le 17 août 2022, Amazon Lex V2 a publié une modification de la façon dont les conversations sont gérées avec l'utilisateur. Cette modification vous permet de mieux contrôler le chemin emprunté par l'utilisateur tout au long de la conversation. Pour plus d'informations, consultez Comprendre la gestion du flux de conversation. Les robots créés avant le 17 août 2022 ne prennent pas en charge les messages d'accrochage par code de dialogue, la définition de valeurs, la configuration des étapes suivantes et l'ajout de conditions.

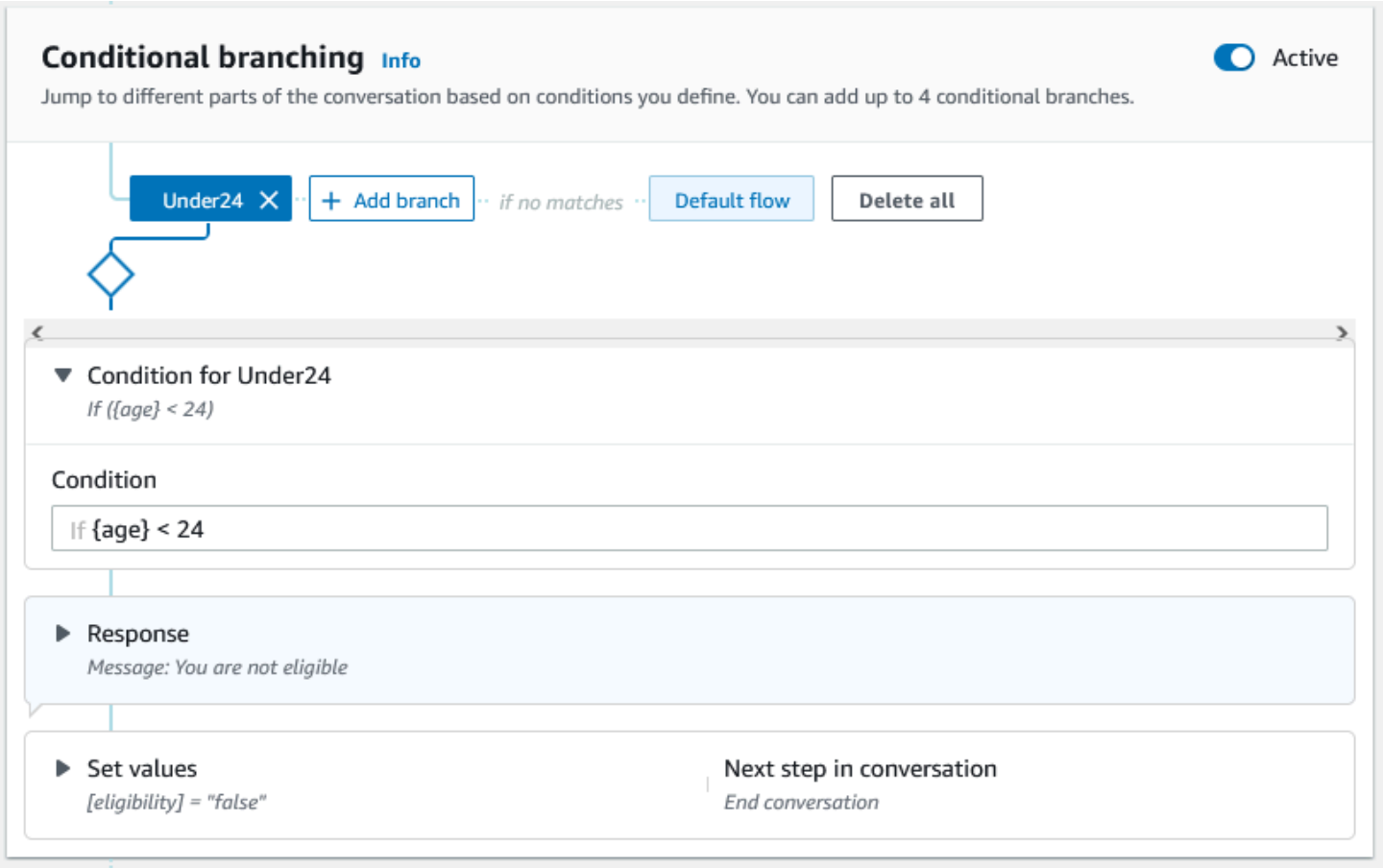
Ajouter des conditions aux conversations dans les succursales

Vous pouvez utiliser le branchement conditionnel pour contrôler le chemin emprunté par votre client tout au long de la conversation avec votre bot. Vous pouvez diviser la conversation en fonction des valeurs des créneaux, des attributs de session, du contenu du mode de saisie et des champs de transcription d'entrée, ou d'une réponse d'Amazon Kendra.

Vous pouvez définir jusqu'à quatre branches. Chaque succursale a une condition qui doit être remplie pour qu'Amazon Lex V2 suive cette branche. Si aucune des branches n'a sa condition satisfaite, une branche par défaut est utilisée.

Lorsque vous définissez une branche, vous définissez l'action qu'Amazon Lex V2 doit entreprendre si les conditions correspondant à cette branche s'avèrent vraies. Vous pouvez définir l'une des actions suivantes :

- Réponse envoyée à l'utilisateur.
- Valeurs des emplacements à appliquer aux emplacements.
- Valeurs des attributs de session pour la session en cours.
- La prochaine étape de la conversation. Pour plus d'informations, consultez Création de parcours de conversation.



Chaque branche conditionnelle possède une expression booléenne qui doit être satisfaite pour qu'Amazon Lex V2 suive la branche. Il existe des opérateurs de comparaison et booléens, des fonctions et des opérateurs de quantification que vous pouvez utiliser pour vos conditions. Par exemple, la condition suivante renvoie true si le créneau {age} est inférieur à 24.

```
{age} < 24
```

La condition suivante renvoie true si le slot à valeurs multiples {toppings} contient le mot « ananas ».

```
{toppings} CONTAINS "pineapple"
```

Vous pouvez combiner plusieurs opérateurs de comparaison avec un opérateur booléen pour des conditions plus complexes. Par exemple, la condition suivante renvoie true si la valeur du slot {make} est « Honda » et que la valeur du slot {model} est « Civic ». Utilisez des parenthèses pour définir l'ordre d'évaluation.

```
({make} = "Honda") AND ({model} = "Civic")
```

Les rubriques suivantes fournissent des informations détaillées sur les opérateurs et les fonctions de branche conditionnels.

> **Note**
>
> Le 17 août 2022, Amazon Lex V2 a publié une modification de la façon dont les conversations sont gérées avec l'utilisateur. Cette modification vous permet de mieux contrôler le chemin emprunté par l'utilisateur tout au long de la conversation. Pour plus d'informations, consultez Comprendre la gestion du flux de conversation. Les robots créés avant le 17 août 2022 ne prennent pas en charge les messages d'accrochage par code de dialogue, la définition de valeurs, la configuration des étapes suivantes et l'ajout de conditions.

Rubriques

Opérateurs de comparaison

Amazon Lex V2 prend en charge les opérateurs de comparaison suivants pour les conditions :

- Est égal à (=)

- Pas égal à (! =)
- Inférieur à (<)
- Inférieur ou égal à (<=)
- supérieure à (>)
- Supérieur ou égal à (>=)

Lorsqu'un opérateur de comparaison est utilisé, il applique les règles suivantes.

- Le côté gauche doit être une référence. Par exemple, pour référencer une valeur d'emplacement, vous utilisez`{slotName}`. Pour référencer une valeur d'attribut de session, vous utilisez`[attribute]`. Pour le mode de saisie et la transcription d'entrée, vous utilisez `$.inputMode` et`$.inputTranscript`.
- Le côté droit doit être une constante et du même type que le côté gauche.
- Toute expression faisant référence à un attribut qui n'a pas été défini est considérée comme non valide et n'est pas évaluée.
- Lorsque vous comparez un emplacement à valeurs multiples, la valeur utilisée est une liste séparée par des virgules de toutes les valeurs interprétées.

Les comparaisons sont basées sur le type de slot de la référence. Elles sont résolues comme suit :

- Chaînes : les chaînes sont comparées en fonction de leur représentation ASCII. Cette comparaison n'est pas sensible à la casse.
- Nombres : les emplacements basés sur des nombres sont convertis de la représentation sous forme de chaîne en un nombre, puis comparés.
- Date/heure : les créneaux temporels sont comparés en fonction de la série chronologique. La date ou l'heure antérieure est considérée comme inférieure. Pour les durées, les périodes plus courtes sont considérées comme plus courtes.

Opérateurs booléens

Amazon Lex V2 prend en charge les opérateurs booléens pour combiner les opérateurs de comparaison. Ils vous permettent de créer des instructions similaires aux suivantes :

```
({number} >= 5) AND ({number} <= 10)
```

Vous pouvez utiliser les opérateurs booléens suivants :

- ET (&&)
- OU (||)
- PAS (!)

Opérateurs de quantification

Les opérateurs de quantification évaluent les éléments d'une séquence et déterminent si un ou plusieurs éléments satisfont à la condition.

- CONTAINS — détermine si la valeur spécifiée est contenue dans un emplacement à valeurs multiples et renvoie la valeur true si c'est le cas. Par exemple, `{toppings} CONTAINS "pineapple"` renvoie true si l'utilisateur a commandé de l'ananas sur sa pizza.

Fonctions

Les fonctions doivent être préfixées par la chaîne`fn.`. L'argument de la fonction est une référence à un emplacement, à un attribut de session ou à un attribut de demande. Amazon Lex V2 fournit deux fonctions permettant d'obtenir des informations à partir des valeurs des slots, de SessionAttribute ou de RequestAttribute.

- fn.Count () — compte le nombre de valeurs dans un emplacement à valeurs multiples.

  Par exemple, si le slot `{toppings}` contient la valeur « pepperoni, ananas » :

  ```
  fn.COUNT({toppings}) = 2
  ```

- fn.is_set () — la valeur est vraie si un emplacement, un attribut de session ou un attribut de demande est défini dans la session en cours.

  Sur la base de l'exemple précédent :

  ```
  fn.IS_SET({toppings})
  ```

- fn.length () — la valeur est la longueur de la valeur de l'attribut de session, de la valeur du slot ou de l'attribut de slot défini dans la session en cours. Cette fonction ne prend pas en charge les emplacements à valeurs multiples ni les emplacements composites.

  Exemple :

Si le slot `{credit-card-number}` contient la valeur « 123456781234 » :

```
fn.LENGTH({credit-card-number}) = 12
```

Exemples d'expressions conditionnelles

Voici quelques exemples d'expressions conditionnelles. REMARQUE : `$.` représente le point d'entrée de la réponse JSON Amazon Lex. La valeur suivante `$.` sera analysée dans la réponse Amazon Lex pour récupérer la valeur. Les expressions conditionnelles utilisant le chemin JSON faisant référence au bloc de transcriptions dans la réponse Amazon Lex ne seront prises en charge que dans les mêmes paramètres régionaux qui prennent en charge les scores de transcription ASR.

Type de la valeur	Cas d'utilisation	Expression conditionnelle
Emplacement personnalisé	`pizzaSize` la valeur de l'emplacement est égale à grande	`{pizzaSize} = "large"`
Emplacement personnalisé	`pizzaSize` est égal à grand ou moyen	`{pizzaSize} = "large"`OU `{pizzaSize} = "medium"`
Emplacement personnalisé	Expressions avec `()` et `AND/OR`	`{pizzaType} = "pepperoni"` OU `{pizzaSize} = "medium"` OU `{pizzaSize} = "small"`
Emplacement personnalisé (emplacement à valeurs multiples)	Vérifiez si l'une des garnitures est de l'oignon	`{toppings} CONTAINS "Onion"`
Emplacement personnalisé (emplacement à valeurs multiples)	Le nombre de garnitures est supérieur à 3	`fn.COUNT({topping}) > 2`

Type de la valeur	Cas d'utilisation	Expression conditionnelle
AMAZON.AlphaNumeric	bookingID est ABC123	{bookingID} = "ABC123"
AMAZON.Number	la valeur de la tranche d'âge est supérieure à 30	{age} > 30
AMAZON.Number	la valeur de la tranche d'âge est égale à 10	{age} = 10
AMAZON.Date	dateOfBirth valeur du créneau avant 1990	{dateOfBirth} < "1990-10-01"
AMAZON.State	destinationState la valeur du slot est égale à Washington	{destinationState} = "washington"
AMAZON.Country	destinationCountry la valeur du slot n'est pas celle des États-Unis	{destinationCountry} != "united states"
AMAZON.FirstName	firstName la valeur du slot est John	{firstName} = "John"
AMAZON.PhoneNumber	phoneNumber la valeur du slot est 716767891932	{phoneNumer} = 716767891932
AMAZON.Percentage	Vérifiez si la valeur du créneau en pourcentage est supérieure ou égale à 78	{percentage} >= 78
AMAZON.EmailAddress	emailAddress la valeur du slot est userA@hmail.com	{emailAddress} = "userA@hmail.com"
AMAZON.LastName	lastNamela valeur du slot est Doe	{lastName} = "Doe"

Type de la valeur	Cas d'utilisation	Expression conditionnelle
`AMAZON.City`	La valeur du slot City est égale à Seattle	`{city} = "Seattle"`
`AMAZON.Time`	Il est après 20 heures	`{time} > "20:00"`
`AMAZON.StreetName`	`streetName` la valeur du slot est Boren Avenue	`{streetName} = "boren avenue"`
`AMAZON.Duration`	`travelDuration` la valeur du slot est inférieure à 2 heures	`{travelDuration} < P2H`
Mode de saisie	Le mode de saisie est vocal	`$.inputMode = "Speech"`
Transcription d'entrée	La transcription d'entrée est égale à « Je veux une grosse pizza »	`$.inputTranscript = "I want a large pizza"`
Attribut de session	vérifier l'attribut customer_subscription_type	`[customer_subcription_type] = "yearly"`
Attribut de demande	vérifier l'indicateur retry_enabled	`((retry_enabled)) = "TRUE"`
Réponse de Kendra	La réponse de Kendra contient une FAQ	`fn.IS_SET(((x-amz-lex:kendra-search-response-question_answer-question-1)))`
Expression conditionnelle avec transcriptions	Expressions conditionnelles utilisant le chemin JSON des transcriptions	`$.transcriptions[0].transcriptionConfidence < 0.8 AND $.transcriptions[1].transcriptionConfidence > 0.5`

Type de la valeur	Cas d'utilisation	Expression conditionnelle
Définir les attributs de session	Définissez les attributs de session à l'aide des transcriptions, du chemin JSON et des valeurs d'emplacement	`[sessionAttribute] = "$.transcriptions...." AND [sessionAttribute] = "{<slotName>}"`
Définir les valeurs des emplacements	Définissez les valeurs des emplacements à l'aide des attributs de session et des transcriptions (chemin JSON)	`{slotName} = [<sessionAttribute>] AND {slotName} = "$.transcriptions...."`

> **Note**
>
> `slotName`fait référence au nom d'un emplacement dans le bot Amazon Lex. Si le slot n'est pas résolu (null), ou s'il n'existe pas, les assignations sont ignorées lors de l'exécution. `sessionAttribute`fait référence au nom de l'attribut de session défini par le client au moment de la création.

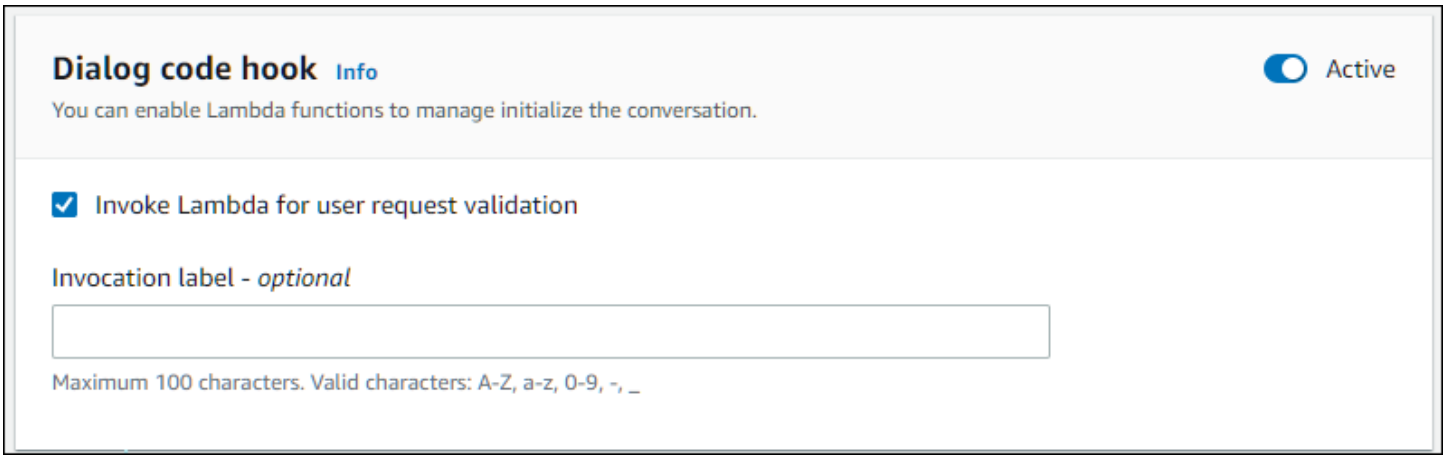
Invoquer le crochet de code de dialogue

À chaque étape de la conversation, lorsqu'Amazon Lex envoie un message à l'utilisateur, vous pouvez utiliser une fonction Lambda comme étape suivante de la conversation. Vous pouvez utiliser cette fonction pour implémenter une logique métier en fonction de l'état actuel de la conversation.

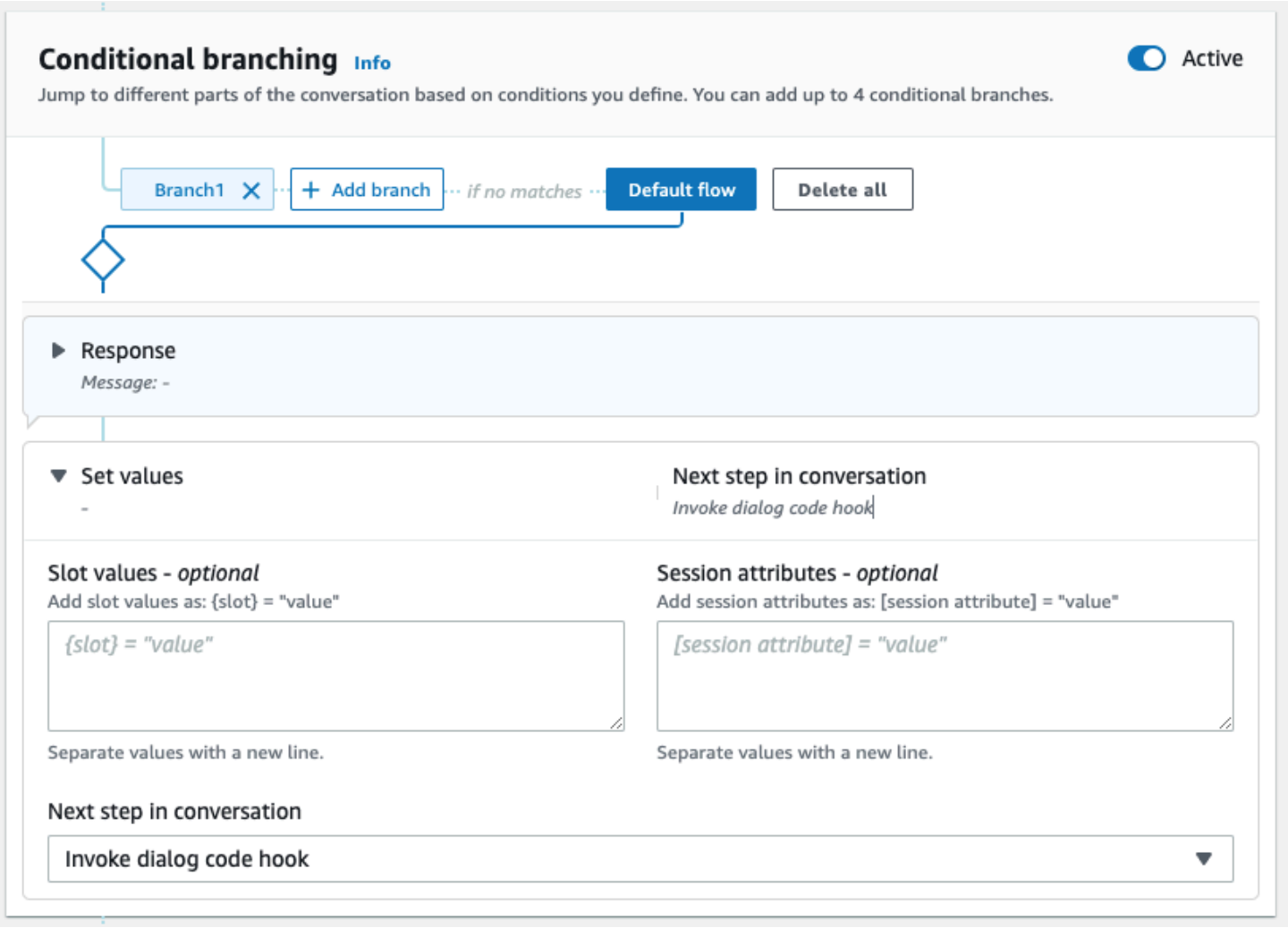
La fonction Lambda qui s'exécute est associée à l'alias du bot que vous utilisez. Pour appeler la fonction Lambda sur tous les crochets de code de dialogue de votre intention, vous devez sélectionner Utiliser une fonction Lambda pour l'initialisation et la validation de l'intention. Pour plus d'informations sur le choix d'une fonction Lambda, consultez. Création et association d'une fonction Lambda à un alias de bot

L'utilisation d'une fonction Lambda comporte deux étapes. Tout d'abord, vous devez activer le code hook du dialogue à tout moment de la conversation. Ensuite, vous devez définir l'étape suivante de la conversation pour utiliser le code hook du dialogue.

L'image suivante montre le crochet codé activé dans la boîte de dialogue.

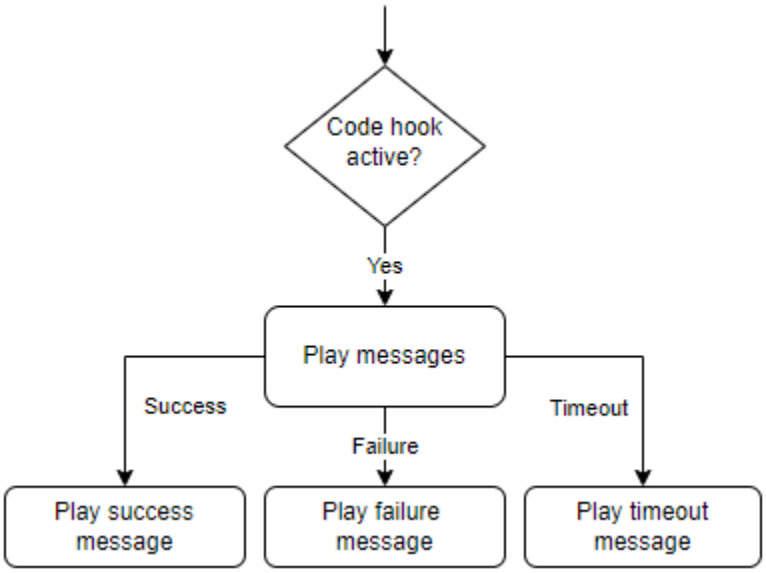


Ensuite, définissez le crochet de code comme action suivante de l'étape de conversation. Vous pouvez le faire en configurant l'étape suivante de la conversation pour Invoke Dialog Code Hook. L'image suivante montre une branche conditionnelle dans laquelle l'appel du crochet de code de dialogue est l'étape suivante pour définir le chemin par défaut de la conversation.



Lorsque les crochets de code sont actifs, vous pouvez définir trois réponses à renvoyer à l'utilisateur :

- Succès — Envoyé lorsque la fonction Lambda s'est terminée avec succès.
- Échec — Envoyé en cas de problème lors de l'exécution de la fonction Lambda ou si la fonction Lambda renvoyait une valeur de. `intent.state` `Failed`
- Timeout — Envoyé si la fonction Lambda ne s'est pas terminée dans le délai d'expiration configuré.



Choisissez Lambda Dialog Code Hook, puis sélectionnez Options avancées pour voir les trois options de réponses correspondant à l'invocation de la fonction Lambda. Vous pouvez définir des valeurs, configurer les étapes suivantes et appliquer des conditions correspondant à chaque réponse pour concevoir le flux de conversation. En l'absence de condition ou d'étape suivante explicite, Amazon Lex V2 décide de l'étape suivante en fonction de l'état actuel de la conversation.

Sur la page des options avancées, vous pouvez également choisir d'activer ou de désactiver l'invocation de votre fonction Lambda. Lorsque la fonction est activée, le crochet du code de dialogue est invoqué avec un appel Lambda, suivi du message de réussite, d'échec ou de délai d'expiration basé sur les résultats de l'appel Lambda. Lorsque la fonction est désactivée, Amazon Lex V2 n'exécute pas la fonction Lambda et procède comme si le crochet du code de dialogue était réussi.

Vous pouvez également définir une étiquette d'appel qui est envoyée à la fonction Lambda lorsqu'elle est invoquée par ce message. Vous pouvez l'utiliser pour identifier la section de votre fonction Lambda à exécuter.

> **Note**
>
> Le 17 août 2022, Amazon Lex V2 a publié une modification de la façon dont les conversations sont gérées avec l'utilisateur. Cette modification vous permet de mieux contrôler le chemin emprunté par l'utilisateur tout au long de la conversation. Pour plus d'informations, consultez Comprendre la gestion du flux de conversation. Les robots créés avant le 17 août 2022 ne prennent pas en charge les messages d'accrochage par code

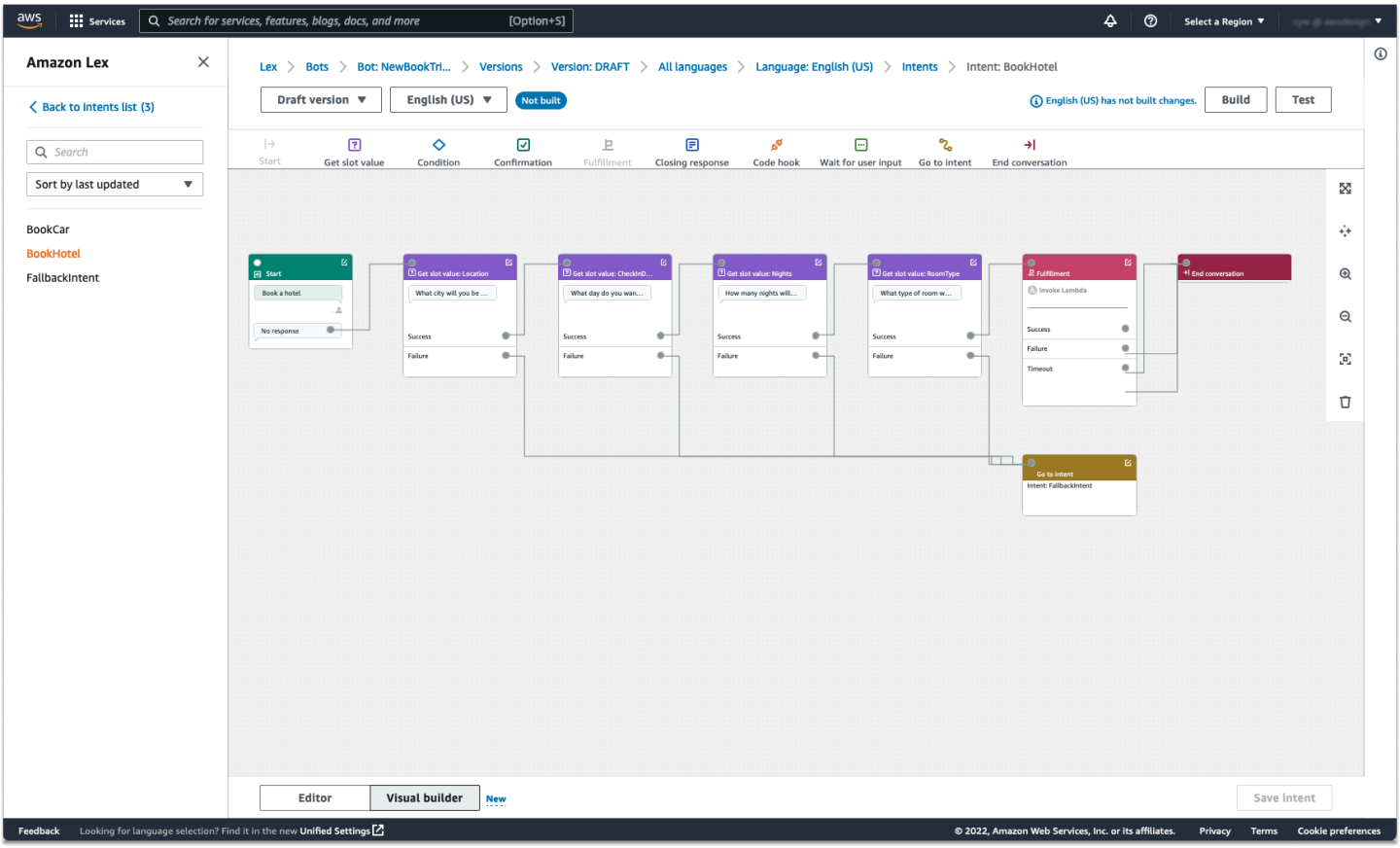
de dialogue, la définition de valeurs, la configuration des étapes suivantes et l'ajout de conditions.

Utilisation du générateur de conversation visuel

Visual Conversation Builder est un générateur de conversation par glisser-déposer qui permet de concevoir et de visualiser facilement des parcours de conversation en utilisant les intentions dans un environnement visuel riche.

Pour accéder au générateur visuel de conversations

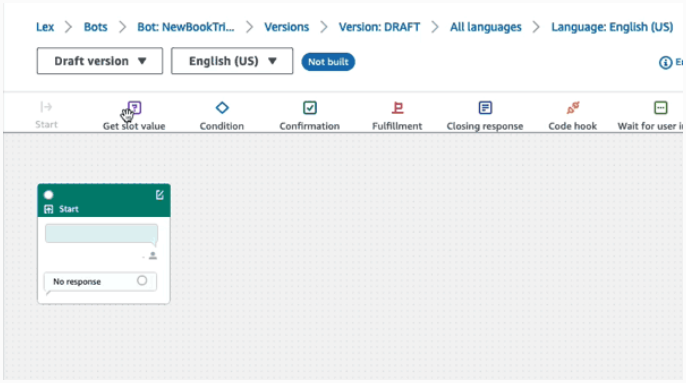
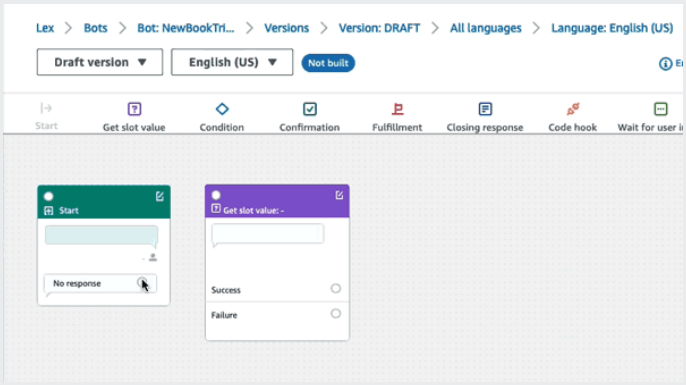
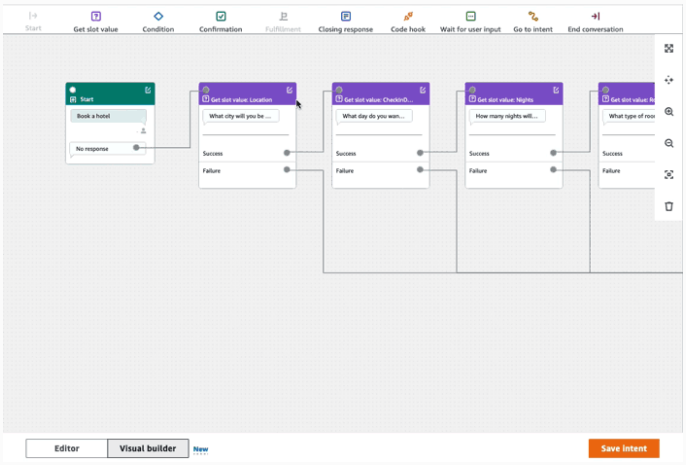
1. Dans la console Amazon Lex V2, choisissez un bot et sélectionnezIntentionsdepuis le volet de navigation de gauche.
2. Accédez à l'éditeur d'intention de l'une des manières suivantes :
   - SélectionnezAjouter une intentiondans le coin supérieur droit duIntentionssection, puis choisissez d'ajouter une intention vide ou une intention intégrée.
   - Choisissez le nom d'une intention dans leIntentionssection.
3. Dans l'éditeur d'intention, sélectionnezGénérateur visueldans le volet en bas de l'écran pour accéder au générateur de conversation visuel.
4. Pour revenir à l'interface de l'éditeur d'intention du menu, sélectionnezRédacteur.

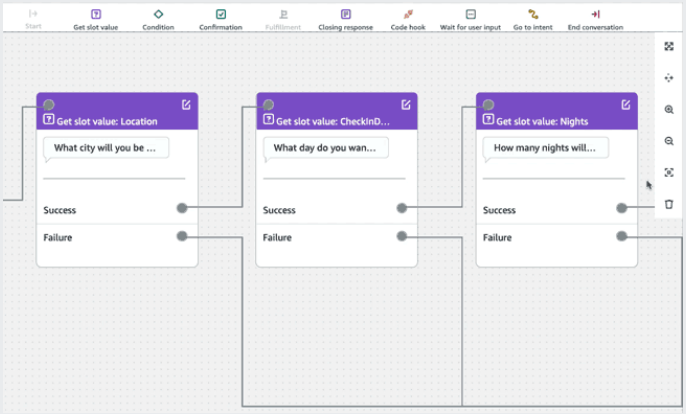
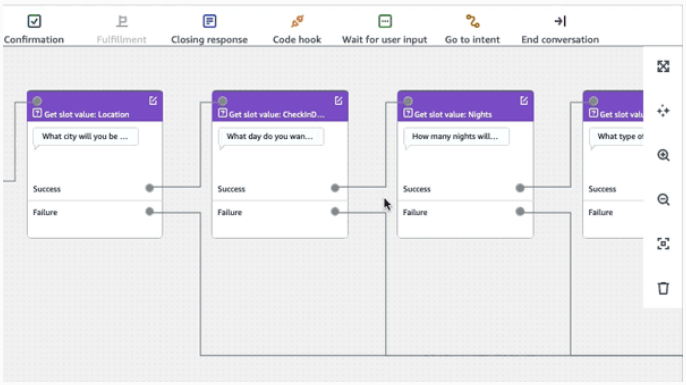
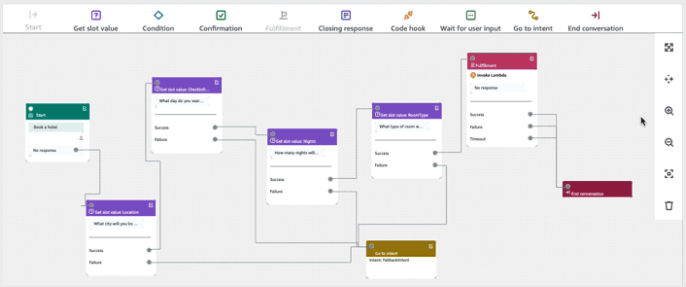


Le générateur visuel de conversations offre une interface utilisateur plus intuitive avec la possibilité de visualiser et de modifier le flux de conversation. En faisant glisser les blocs, vous pouvez étendre un flux existant ou réorganiser les étapes de la conversation. Vous pouvez développer un flux de conversation avec des branchements complexes sans écrire de code Lambda.

Cette modification permet de découpler la conception du flux de conversation des autres logiques métier dans Lambda. Le générateur visuel de conversation peut être utilisé conjointement avec l'éditeur d'intention existant et peut être utilisé pour créer des flux de conversation. Il est toutefois recommandé d'utiliser la vue de l'éditeur visuel pour les flux de conversation plus complexes.

Lorsque vous enregistrez une intention, Amazon Lex V2 peut connecter automatiquement les intentions lorsqu'il détermine qu'il y a des connexions manquées, qu'Amazon Lex V2 suggère une connexion ou que vous pouvez sélectionner votre propre connexion pour le bloc.

Action	Exemple
Ajouter un bloc à l'espace de travail	
Etablissement d'une connexion entre les blocs	
Ouvrir le panneau de configuration sur un bloc	

Action	Exemple
Zoomez pour ajuster	
Supprimer un bloc du flux de conversation	
Nettoyage automatique de l'espace de travail	

Terminologie :

Bloquer— L'unité de base d'un flux de conversation. Chaque bloc possède une fonctionnalité spécifique permettant de gérer différents cas d'utilisation d'une conversation.

Porto— Chaque bloc contient des ports qui peuvent être utilisés pour connecter un bloc à un autre. Les blocs peuvent contenir des ports d'entrée et des ports de sortie. Chaque port de sortie représente

une variation fonctionnelle particulière d'un bloc (telle que des erreurs, des délais d'attente ou un succès).

Bord— Une arête est une connexion entre le port de sortie d'un bloc et le port d'entrée d'un autre bloc. Il fait partie d'une branche d'un flux de conversation.

Flux de conversation— Un ensemble de blocs reliés par des arêtes qui décrit les interactions au niveau de l'intention avec un client.

Blocs

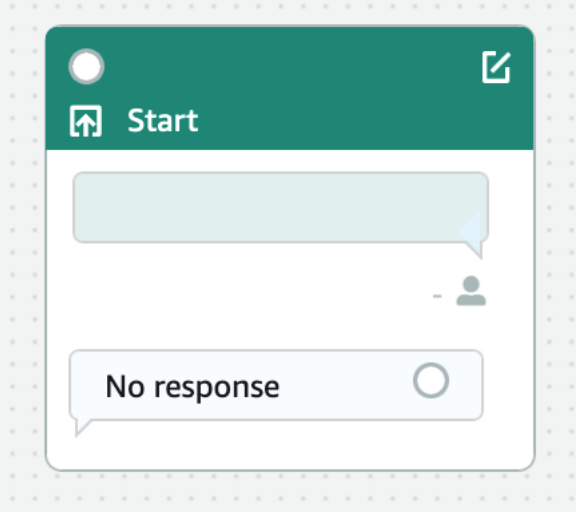
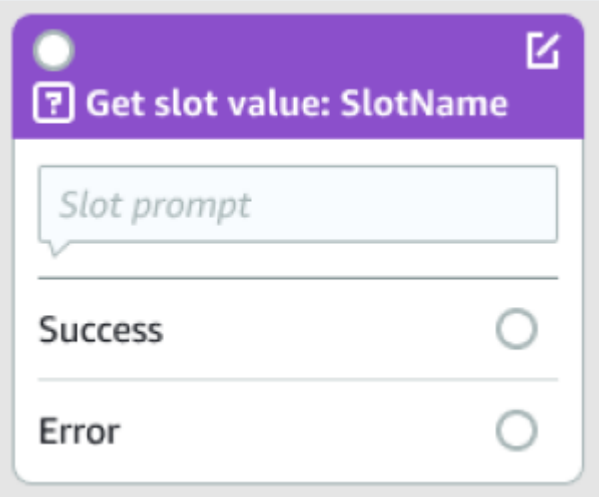
Les blocs sont les éléments constitutifs de la conception d'un flux de conversation. Ils représentent différents états de l'intention, depuis le début de l'intention jusqu'à la fin, en passant par la saisie par l'utilisateur.

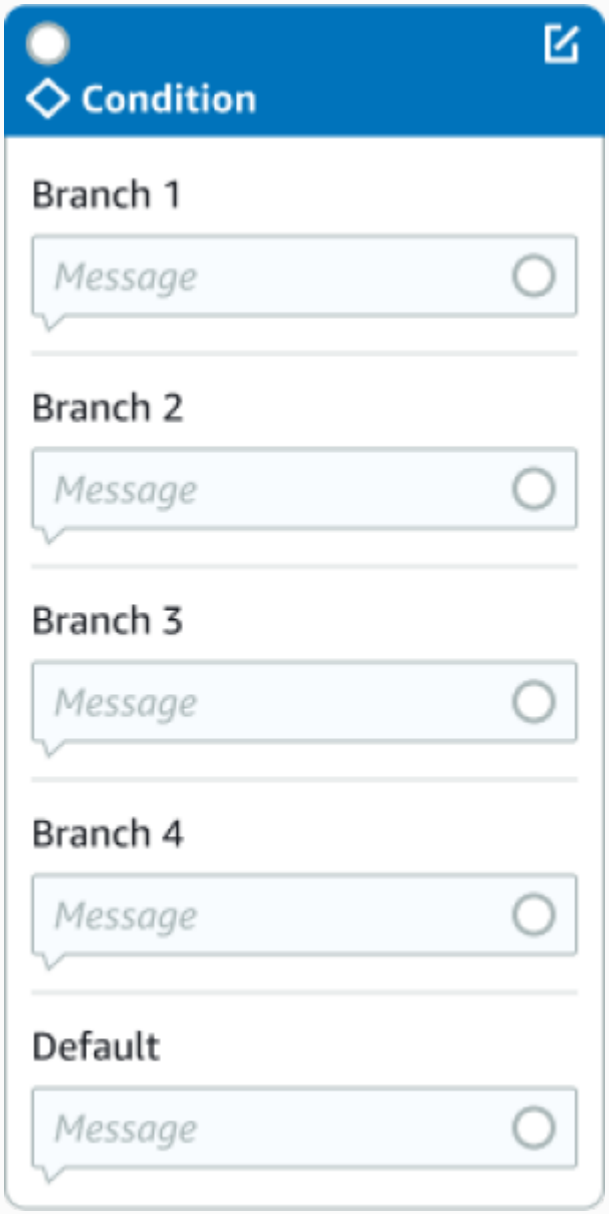
Chaque bloc possède un point d'entrée et un ou plusieurs points de sortie en fonction du type de bloc. Chaque point de sortie peut être configuré avec un message correspondant au fur et à mesure que la conversation passe par les points de sortie. Pour les blocs comportant plusieurs points de sortie, les points de sortie sont liés à l'état correspondant au nœud. Pour un nœud de condition, les points de sortie représentent les différentes conditions.

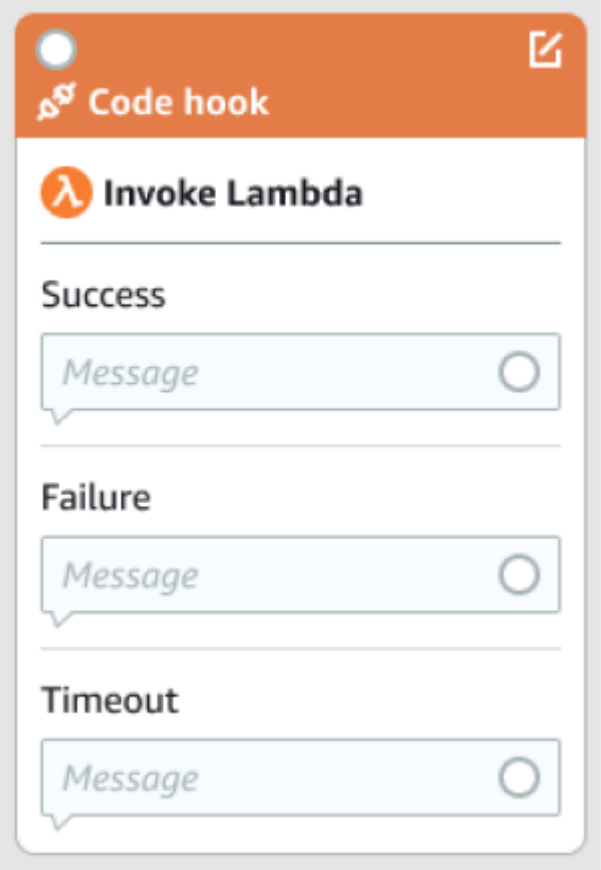
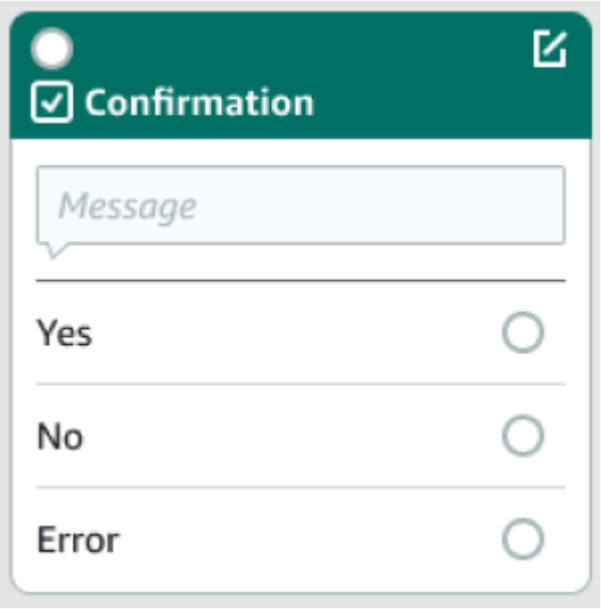
Chaque bloc possède un panneau de configuration, qui s'ouvre en cliquant surModifiericône dans le coin supérieur droit du bloc. Le panneau de configuration contient des champs détaillés qui peuvent être configurés pour correspondre à chaque bloc.

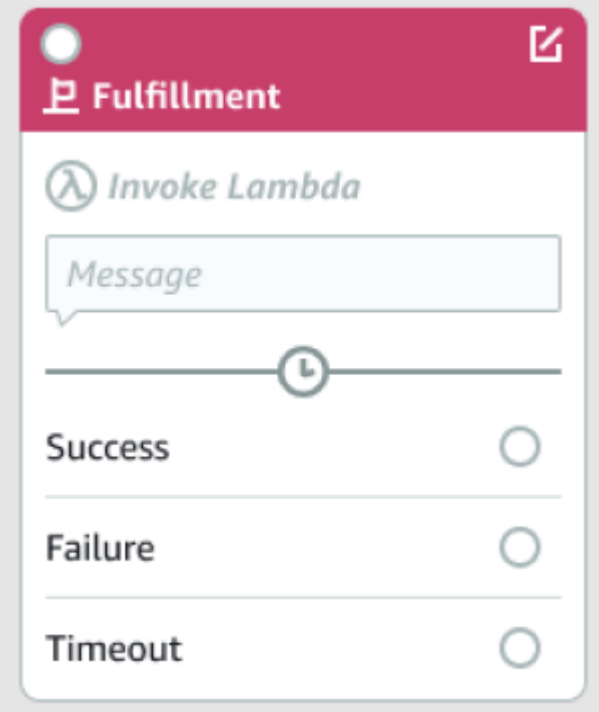
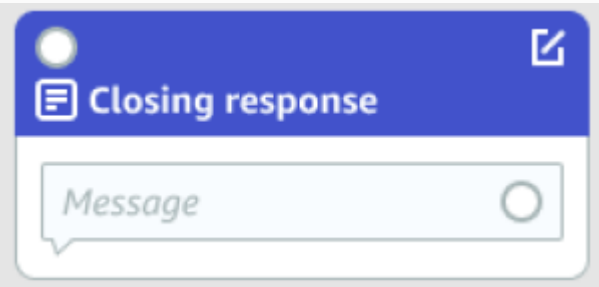
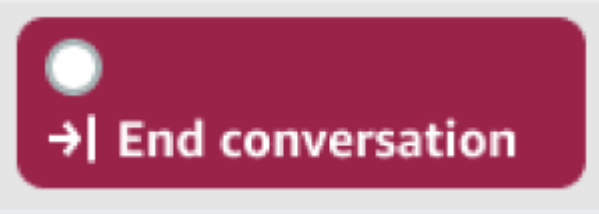
Les invites et les messages du bot peuvent être configurés directement sur le nœud en faisant glisser un nouveau bloc, ou ils peuvent être modifiés dans le panneau de droite, ainsi que d'autres attributs du bloc.

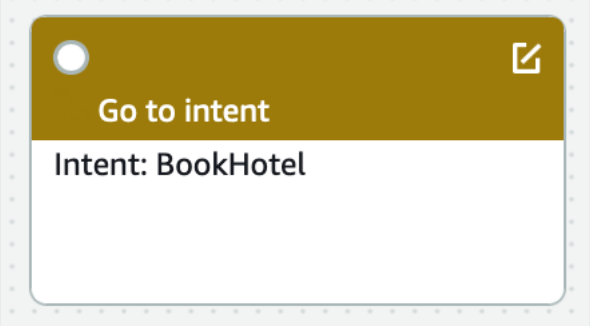
Types de blocs— Voici les types de blocs que vous pouvez utiliser avec Visual Conversation Builder.

Type de bloc	Block
Démarrer— La racine ou le premier bloc du flux de conversation. Ce bloc peut également être configuré de telle sorte que le bot puisse envoyer une réponse initiale (message indiquant que l'intention a été reconnue). Pour plus d'informations, veuillez consulter Réponse initiale.	
Obtenir la valeur de l'emplacement— Ce bloc essaie d'obtenir une valeur pour un seul emplacement. Ce bloc comporte un paramètre permettant d'attendre la réponse du client à l'invite de demande de créneau. Pour plus d'informations, veuillez consulter Emplacements.	

Type de bloc	Block
État— Ce bloc contient des conditions. Il contient jusqu'à 4 branches personnalisées (avec conditions) et une branche par défaut. Pour plus d'informations, veuillez consulter Ajouter des conditions aux conversations dans les succursales.	

Type de bloc	Block
Crochet de code de dialogue— Ce bloc gère l'invocation de la fonction Lambda de la boîte de dialogue. Ce bloc contient les réponses des robots basées sur la réussite, l'échec ou l'expiration du délai de la fonction Lambda de la boîte de dialogue. Pour plus d'informations, veuillez consulter Invoquer le crochet de code de dialogue.	
Confirmation— Ce bloc interroge le client avant que l'intention ne soit réalisée. Il contient des réponses de robots basées sur le fait que le client dit oui ou non à l'invite de confirmation. Pour plus d'informations, veuillez consulter Confirmation.	

Type de bloc	Block
Exécution— Ce bloc gère la réalisation de l'intention, généralement après l'obtention des créneaux. Il peut être configuré pour invoquer des fonctions Lambda et pour répondre par des messages, en cas de réussite ou d'échec de l'exécution. Pour plus d'informations, veuillez consulter Exécution.	
Réponse finale— Ce bloc permet au bot de répondre par un message avant de mettre fin à la conversation. Pour plus d'informations, veuillez consulter Réponse finale.	
Mettre fin à la conversation— Ce bloc indique la fin du flux de conversation.	
Attendre la saisie de l'utilisateur— Ce bloc peut être utilisé pour capturer les informations du client et passer à une autre intention en fonction de l'énoncé.	

Type de bloc	Block
Accéder à l'intention— Ce bloc peut être utilisé pour accéder à une nouvelle intention ou pour obtenir directement un créneau spécifique de cette intention.	

Types de ports

Tous les blocs contiennent un port d'entrée, qui est utilisé pour connecter ses blocs parents. La conversation ne peut circuler vers le port d'entrée d'un bloc particulier qu'à partir du port de sortie de son bloc parent. Toutefois, les blocs peuvent contenir zéro, un ou plusieurs ports de sortie. Les blocs dépourvus de ports de sortie indiquent la fin du flux de conversation dans l'intention actuelle (`GoToIntent`,`EndConversation`,`WaitForUserInput`).

Règles de conception des intentions :

- Tous les flux d'une intention commencent par le bloc de départ.
- Les messages correspondant à chaque point de sortie sont facultatifs.
- Vous pouvez configurer les blocs pour définir des valeurs correspondant à chaque point de sortie dans le panneau de configuration.
- Il ne peut exister qu'un seul bloc de démarrage, de confirmation, d'exécution et de clôture dans un même flux au sein d'une intention. Des conditions multiples, un crochet par code de boîte de dialogue, l'obtention des valeurs des emplacements, la fin de la conversation, le transfert et l'attente des blocs de saisie de l'utilisateur peuvent exister.
- Un bloc de conditions ne peut pas avoir de connexion directe à un bloc de conditions. Il en va de même pour le crochet de code de boîte de dialogue.
- Trois blocs sont autorisés pour les flux circulaires, mais aucun connecteur entrant vers Start Intent n'est autorisé.
- Un emplacement optionnel ne possède pas de connecteur entrant ni de connexion sortante et est principalement utilisé pour capturer toutes les données présentes lors de la sollicitation de l'intention. Tout autre créneau faisant partie du chemin de conversation doit être un créneau obligatoire.

Blocs :

- Le bloc de départ doit avoir un bord sortant.
- Chaque bloc de valeur du slot get doit avoir un bord sortant du port de réussite, si le slot est requis.
- Chaque bloc de condition doit avoir une arête sortante depuis chaque branche si le bloc est actif.
- Un bloc de conditions ne peut pas avoir plus d'un parent.
- Un bloc conditionnel actif doit avoir une arête entrante.
- Chaque bloc de crochet de code actif doit avoir un bord sortant depuis chaque port : succès, échec et délai d'expiration.
- Un bloc de crochet de code actif doit avoir un bord entrant.
- Un bloc de confirmation actif doit avoir un bord entrant.
- Un bloc de distribution actif doit comporter un bord entrant.
- Un bloc de fermeture actif doit avoir une arête entrante.
- Un bloc de conditions doit comporter au moins une branche autre que celle par défaut.
- Une intention doit être spécifiée pour un bloc d'accès à l'intention.

Bords :

- Un bloc de conditions ne peut pas être connecté à un autre bloc de conditions.
- Un bloc de crochet de code ne peut pas être connecté à un autre bloc de crochet de code.
- Un bloc de conditions ne peut être connecté qu'à zéro ou à un seul bloc de crochet de code.
- La connexion (crochet de code -> condition -> crochet de code) n'est pas valide.
- Un bloc de distribution ne peut pas comporter de bloc de crochet à code lorsqu'il est enfant.
- Un bloc de conditions, qui est un enfant du bloc de distribution, ne peut pas avoir d'enfant de bloc de crochet de code.
- Un bloc de fermeture ne peut pas comporter de bloc de crochet codé lorsqu'il est enfant.
- Un bloc de conditions qui est un enfant du bloc de fermeture ne peut pas avoir de bloc de crochet de code enfant.
- Un bloc de valeur d'emplacement de démarrage, de confirmation ou de saisie ne peut comporter qu'un seul bloc de crochet de code dans sa chaîne de dépendances.

> **Note**
>
> Le 17 août 2022, Amazon Lex V2 a publié une modification de la façon dont les conversations sont gérées avec l'utilisateur. Cette modification vous permet de mieux contrôler le chemin emprunté par l'utilisateur tout au long de la conversation. Pour plus d'informations, veuillez consulter Comprendre la gestion du flux de conversation. Les robots créés avant le 17 août 2022 ne prennent pas en charge les messages d'accroche à code de dialogue, la définition de valeurs, la configuration des étapes suivantes et l'ajout de conditions.

Intentions prédéfinies

Pour les actions courantes, vous pouvez utiliser la bibliothèque d'intentions intégrée standard. Pour créer une intention à partir d'une intention prédéfinie, choisissez une intention prédéfinie dans la console et attribuez-lui un nouveau nom. La nouvelle intention a la même configuration que l'intention de base, telle que les exemples d'énoncés.

Dans l'implémentation actuelle, vous ne pouvez pas effectuer les actions suivantes :

- Ajouter ou supprimer des exemples d'énoncés dans l'intention de base
- Configurer les options pour les intentions prédéfinies

Pour ajouter une intention intégrée à un bot

1. Connectez-vous à la console Amazon Lex AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/lex/.
2. Choisissez le bot auquel ajouter l'intention intégrée.
3. Dans le menu de gauche, choisissez la langue, puis sélectionnez Intents.
4. Choisissez Ajouter une intention, puis choisissez Utiliser une intention intégrée.
5. Dans Intention intégrée, choisissez l'intention à utiliser.
6. Donnez un nom à l'intention, puis choisissez Ajouter.
7. Utilisez l'éditeur d'intention pour configurer l'intention selon les besoins de votre bot.

Rubriques

- AMAZON.CancelIntent
- AMAZON.FallbackIntent
- AMAZON.HelpIntent
- AMAZON.KendraSearchIntent
- AMAZON.PauseIntent
- AMAZON.QnAIntent
- AMAZON.RepeatIntent
- AMAZON.ResumeIntent
- AMAZON.StartOverIntent
- AMAZON.StopIntent

AMAZON.CancelIntent

Répond aux mots et aux phrases qui indiquent que l'utilisateur souhaite annuler l'interaction en cours. Votre application peut utiliser cette intention pour supprimer les valeurs de type d'emplacement et d'autres attributs avant de mettre fin à l'interaction avec l'utilisateur.

Énoncés courants :

- annuler
- tant pis
- Oubliez ça

AMAZON.FallbackIntent

Lorsque la saisie d'une intention par un utilisateur ne correspond pas aux attentes d'un bot, vous pouvez configurer Amazon Lex V2 pour qu'il invoque une intention de secours. Par exemple, si la saisie par l'utilisateur « J'aimerais commander des bonbons » ne correspond pas à l'intention de votre `OrderFlowers` bot, Amazon Lex V2 invoque l'intention de secours pour gérer la réponse.

Le type `AMAZON.FallbackIntent` d'intention intégré est automatiquement ajouté à votre bot lorsque vous créez un bot à l'aide de la console ou lorsque vous ajoutez des paramètres régionaux à un bot à l'aide de l'CreateBotLocaleopération.

L'appel d'une intention de secours se fait en deux étapes. Dans la première étape, l'intention de secours est mise en correspondance en fonction de l'entrée de l'utilisateur. Lorsque l'intention de

secours est mise en correspondance, le comportement du bot dépend du nombre de nouvelles tentatives configurées pour une invite.

Amazon Lex V2 répond à l'objectif de repli dans les situations suivantes :

- L'entrée de l'utilisateur pour une intention ne correspond pas à l'entrée attendue par le bot
- L'entrée audio est du bruit ou l'entrée de texte n'est pas reconnue en tant que mots.
- La saisie de l'utilisateur est ambiguë et Amazon Lex V2 ne peut pas déterminer l'intention à invoquer.

L'intention de secours est appelée lorsque :

- Une intention ne reconnaît pas l'entrée utilisateur comme valeur d'option après le nombre de tentatives configuré.
- Une intention ne reconnaît pas l'entrée utilisateur comme réponse à une invite de confirmation après le nombre de tentatives configuré.

Vous ne pouvez pas ajouter les éléments suivants à une intention de secours :

- Énoncés
- Emplacements
- Une invite de confirmation

Utilisation d'une fonction Lambda avec une intention de repli

Lorsqu'une intention de secours est appelée, la réponse dépend de la valeur du paramètre `fulfillmentCodeHook` définie sur l'opération CreateIntent. Le bot effectue l'une des opérations suivantes :

- Il renvoie les informations d'intention à l'application cliente.
- Appelle la fonction Lambda de validation et d'exécution des alias. Il appelle la fonction avec les variables de session définies pour la session.

Pour plus d'informations sur la définition de la réponse lorsqu'une intention de secours est appelée, consultez le paramètre `fulfillmentCodeHook` de l'opération CreateIntent.

Si vous utilisez la fonction Lambda avec votre intention de secours, vous pouvez utiliser cette fonction pour appeler une autre intention ou pour établir une forme de communication avec l'utilisateur, telle que la collecte d'un numéro de rappel ou l'ouverture d'une session avec un représentant du service client.

Une intention de secours peut être appelée plusieurs fois dans la même session. Supposons, par exemple, que votre fonction Lambda utilise l'action `ElicitIntent` de dialogue pour demander à l'utilisateur une intention différente. Si Amazon Lex V2 ne parvient pas à déduire l'intention de l'utilisateur après le nombre d'essais configuré, il invoque à nouveau l'intention de secours. Il appelle également l'intention de secours lorsque l'utilisateur ne répond pas avec une valeur d'option valide après le nombre de tentatives configuré.

Vous pouvez configurer votre fonction Lambda pour suivre le nombre de fois que l'intention de secours est appelée à l'aide d'une variable de session. Votre fonction Lambda peut effectuer une action différente si elle est appelée plus de fois que le seuil que vous avez défini dans votre fonction Lambda. Pour plus d'informations sur les variables de session, consultez Définition des attributs de session.

AMAZON.HelpIntent

Répond aux mots ou aux phrases qui indiquent que l'utilisateur a besoin d'aide lorsqu'il interagit avec votre bot. Lorsque cette intention est invoquée, vous pouvez configurer votre fonction ou application Lambda pour fournir des informations sur les capacités de votre bot, poser des questions complémentaires sur les domaines d'aide ou confier l'interaction à un agent humain.

Énoncés courants :

- aide
- aidez-moi
- peux-tu m'aider

AMAZON.KendraSearchIntent

Pour rechercher des documents que vous avez indexés avec Amazon Kendra, utilisez l'`AMAZON.KendraSearchIntent`intention. Lorsqu'Amazon Lex V2 ne parvient pas à déterminer l'action suivante dans une conversation avec l'utilisateur, cela déclenche l'intention de recherche.

`AMAZON.KendraSearchIntent`Il est disponible uniquement dans la région anglaise (États-Unis) (en-États-Unis) et dans les régions USA Est (Virginie du Nord), USA Ouest (Oregon) et Europe (Irlande).

Amazon Kendra est un service de machine-learning-based recherche qui indexe les documents en langage naturel tels que les documents PDF ou les fichiers Microsoft Word. Il peut effectuer des recherches sur des documents indexés et renvoyer les types de réponse suivants à une question :

- Une réponse
- Une entrée de FAQ qui pourrait répondre à la question
- Un document lié à la question

Pour obtenir un exemple d'utilisation de `AMAZON.KendraSearchIntent`, veuillez consulter Exemple : création d'un bot FAQ pour un index Amazon Kendra.

Si vous configurez une `AMAZON.KendraSearchIntent` intention pour votre bot, Amazon Lex V2 appelle l'intention chaque fois qu'il ne parvient pas à déterminer l'intention exprimée par l'utilisateur. En l'absence de réponse de la part d'Amazon Kendra, la conversation se poursuit telle que configurée dans le bot.

> **Note**
>
> Amazon Lex V2 ne prend actuellement pas en charge le processus `AMAZON.KendraSearchIntent` d'obtention de créneaux. Si Amazon Lex V2 ne parvient pas à déterminer l'énoncé de l'utilisateur pour un emplacement, il appelle le. `AMAZON.FallbackIntent`

Lorsque vous utilisez le `AMAZON.KendraSearchIntent` with `AMAZON.FallbackIntent` dans le même bot, Amazon Lex V2 utilise les intentions suivantes :

1. Amazon Lex V2 appelle le`AMAZON.KendraSearchIntent`. L'objectif est l'opération Amazon Kendra`Query`.
2. Si Amazon Kendra renvoie une réponse, Amazon Lex V2 affiche le résultat à l'utilisateur.
3. En l'absence de réponse de la part d'Amazon Kendra, Amazon Lex V2 réinvite l'utilisateur. L'action suivante dépend de la réponse de l'utilisateur.

- Si la réponse de l'utilisateur contient un énoncé reconnu par Amazon Lex V2, tel que le remplissage d'une valeur de créneau ou la confirmation d'une intention, la conversation avec l'utilisateur se déroule comme configuré pour le bot.
- Si la réponse de l'utilisateur ne contient aucun énoncé reconnu par Amazon Lex V2, Amazon Lex V2 lance un autre appel à l'`Query`opération.

4. S'il n'y a aucune réponse après le nombre de tentatives configuré, Amazon Lex V2 appelle l'utilisateur `AMAZON.FallbackIntent` et met fin à la conversation avec l'utilisateur.

Vous pouvez utiliser le pour envoyer une demande `AMAZON.KendraSearchIntent` à Amazon Kendra de trois manières :

- Laissez l'intention de recherche faire la demande pour vous. Amazon Lex V2 appelle Amazon Kendra en utilisant l'énoncé de l'utilisateur comme chaîne de recherche. Lorsque vous créez l'intention, vous pouvez définir une chaîne de filtre de requête qui limite le nombre de réponses renvoyées par Amazon Kendra. Amazon Lex V2 utilise le filtre dans la demande de requête.
- Ajoutez des paramètres de requête supplémentaires à la demande pour affiner les résultats de recherche à l'aide de votre fonction Lambda. Vous ajoutez un `kendraQueryFilterString` champ contenant les paramètres de requête Amazon Kendra à l'action de `delegate` dialogue. Lorsque vous ajoutez des paramètres de requête à la demande à l'aide de la fonction Lambda, ils ont priorité sur le filtre de requête que vous avez défini lors de la création de l'intention.
- Créez une nouvelle requête à l'aide de la fonction Lambda. Vous pouvez créer une demande de requête Amazon Kendra complète envoyée par Amazon Lex V2. Vous spécifiez la requête dans le champ `kendraQueryRequestPayload` de l'action de dialogue `delegate`. Le champ `kendraQueryRequestPayload` a priorité sur le champ `kendraQueryFilterString`.

Pour spécifier le `queryFilterString` paramètre lorsque vous créez un bot, ou pour spécifier le `kendraQueryFilterString` champ lorsque vous appelez l'`delegate`action dans une fonction Lambda de dialogue, vous devez spécifier une chaîne qui est utilisée comme filtre d'attribut pour la requête Amazon Kendra. Si la chaîne n'est pas un filtre d'attribut valide, vous obtiendrez une exception `InvalidBotConfigException` lors de l'exécution. Pour plus d'informations sur les filtres d'attributs, consultez la section Utilisation des attributs de document pour filtrer les requêtes dans le manuel Amazon Kendra Developer Guide.

Pour contrôler la requête qu'Amazon Lex V2 envoie à Amazon Kendra, vous pouvez spécifier une requête dans le `kendraQueryRequestPayload` champ de votre fonction Lambda. Si la requête n'est pas valide, Amazon Lex V2 renvoie une `InvalidLambdaResponseException` exception.

Pour plus d'informations, consultez l'opération de requête dans le manuel Amazon Kendra Developer Guide.

Pour obtenir un exemple de la façon d'utiliser l'intention `AMAZON.KendraSearchIntent`, veuillez consulter Exemple : création d'un bot FAQ pour un index Amazon Kendra.

Politique IAM pour Amazon Kendra Search

Pour utiliser l'`AMAZON.KendraSearchIntent`intention, vous devez utiliser un rôle qui fournit des politiques AWS Identity and Access Management (IAM) permettant à Amazon Lex V2 d'assumer un rôle d'exécution autorisé à appeler l'intention Amazon `Query` Kendra. Les paramètres IAM que vous utilisez varient selon que vous les avez créés à l'`AMAZON.KendraSearchIntent`aide de la console Amazon Lex V2, d'un SDK AWS ou du AWS Command Line Interface ()AWS CLI. Lorsque vous utilisez la console, vous pouvez choisir entre ajouter l'autorisation d'appeler Amazon Kendra au rôle lié au service Amazon Lex V2 ou utiliser un rôle spécifique pour appeler l'opération Amazon Kendra. `Query` Lorsque vous utilisez le AWS CLI ou un SDK pour créer l'intention, vous devez utiliser un rôle spécifique pour appeler l'`Query`opération.

Attachement d'autorisations

Vous pouvez utiliser la console pour associer des autorisations d'accès à l'`Query`opération Amazon Kendra au rôle lié au service Amazon Lex V2 par défaut. Lorsque vous associez des autorisations au rôle lié au service, vous n'avez pas besoin de créer et de gérer un rôle d'exécution spécifiquement pour vous connecter à l'index Amazon Kendra.

L'utilisateur, le rôle ou le groupe que vous utilisez pour accéder à la console Amazon Lex V2 doit disposer des autorisations nécessaires pour gérer les politiques relatives aux rôles. Associez la politique IAM suivante au rôle d'accès à la console. Lorsque vous accordez ces autorisations, le rôle dispose des autorisations permettant de modifier la stratégie de rôle liée à un service existante.

```
{
"Version": "2012-10-17",
"Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "iam:AttachRolePolicy",
            "iam:PutRolePolicy",
            "iam:GetRolePolicy"
        ],
```

```
        "Resource": "arn:aws:iam::*:role/aws-service-role/lexv2.amazonaws.com/
AWSServiceRoleForLexBots*"
    },
    {
        "Effect": "Allow",
        "Action": "iam:ListRoles",
        "Resource": "*"
    }
]
}
```

Spécification d'un rôle

Vous pouvez utiliser la console AWS CLI, le ou l'API pour spécifier un rôle d'exécution à utiliser lors de l'appel de l'opération Amazon Kendra`Query`.

L'utilisateur, le rôle ou le groupe que vous utilisez pour spécifier le rôle d'exécution doit disposer de l'`iam:PassRole`autorisation. La stratégie suivante définit l'autorisation. Vous pouvez utiliser les clés de contexte de condition `iam:AssociatedResourceArn` et `iam:PassedToService` pour limiter davantage la portée des autorisations. Pour plus d'informations, consultez les sections IAM et AWS STS Condition Context Keys dans le guide de l'AWS Identity and Access Management utilisateur.

```
{
"Version": "2012-10-17",
"Statement": [
    {
        "Effect": "Allow",
        "Action": "iam:PassRole",
        "Resource": "arn:aws:iam::account:role/role"
    }
]
}
```

Le rôle d'exécution qu'Amazon Lex V2 doit utiliser pour appeler Amazon Kendra doit disposer des `kendra:Query` autorisations requises. Lorsque vous utilisez un rôle IAM existant pour obtenir l'autorisation d'appeler l'opération Amazon `Query` Kendra, le rôle doit être associé à la politique suivante.

Vous pouvez utiliser la console IAM, l'API IAM ou le AWS CLI pour créer une politique et l'associer à un rôle. Ces instructions utilisent l'AWS CLI pour créer le rôle et les stratégies.

> **Note**
>
> Le code suivant est formaté pour Linux et macOS. Sous Windows, remplacez le caractère de continuité de ligne Linux (\) par le caret (^).

Pour ajouter une autorisation d'opération Query à un rôle

1. Créez un document appelé **`KendraQueryPolicy.json`** dans le répertoire courant, ajoutez-y le code suivant et enregistrez-le.

```
{
"Version": "2012-10-17",
"Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "kendra:Query"
        ],
        "Resource": [
            "arn:aws:kendra:region:account:index/index ID"
        ]
    }
]
}
```

2. Dans le AWS CLI, exécutez la commande suivante pour créer la politique IAM permettant d'exécuter l'opération Amazon `Query` Kendra.

```
aws iam create-policy \
--policy-name query-policy-name \
--policy-document file://KendraQueryPolicy.json
```

3. Attachez la politique au rôle IAM que vous utilisez pour appeler l'`Query`opération.

```
aws iam attach-role-policy \
--policy-arn arn:aws:iam::account-id:policy/query-policy-name
--role-name role-name
```

Vous pouvez choisir de mettre à jour le rôle lié au service Amazon Lex V2 ou d'utiliser un rôle que vous avez créé lors de la création du rôle `AMAZON.KendraSearchIntent` pour votre bot. La procédure suivante indique comment choisir le rôle IAM à utiliser.

Pour spécifier le rôle d'exécution pour AMAZON.KendraSearchIntent

1. Connectez-vous à la console Amazon Lex AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/lex/.
2. Choisissez le bot auquel vous souhaitez ajouter l'intention `AMAZON.KendraSearchIntent`.
3. Choisissez le signe plus (+) en regarde de Intents (Intentions).
4. Dans Add intent (Ajouter une intention), choisissez Search existing intents (Rechercher des intentions existantes).
5. Dans Search intents (Rechercher des intentions), entrez, **`AMAZON.KendraSearchIntent`** puis choisissez Add (Ajouter).
6. Dans Copy built-in intent (Copier une intention intégrée), entrez un nom pour l'intention, par exemple **`KendraSearchIntent`**, puis choisissez Add (Ajouter).
7. Ouvrez la section Amazon Kendra query (Requête Amazon Kendra).
8. Pour IAM role (Rôle IAM) choisissez une des options suivantes :

   - Pour mettre à jour le rôle lié au service Amazon Lex V2 afin de permettre à votre bot d'interroger les index Amazon Kendra, choisissez Ajouter des autorisations Amazon Kendra.
   - Pour utiliser un rôle autorisé à appeler l'`Query`opération Amazon Kendra, choisissez Utiliser un rôle existant.

Utilisation des attributs de demande et de session en tant que filtres

Pour filtrer la réponse d'Amazon Kendra aux éléments liés à la conversation en cours, utilisez les attributs de session et de demande comme filtres en ajoutant le `queryFilterString` paramètre lorsque vous créez votre bot. Vous spécifiez un espace réservé pour l'attribut lorsque vous créez l'intention, puis Amazon Lex V2 remplace une valeur avant d'appeler Amazon Kendra. Pour de plus amples informations sur les attributs de demande, veuillez consulter Définition des attributs de demande. Pour en savoir plus sur les attributs de session, consultez Définition des attributs de session.

Voici un exemple de `queryFilterString` paramètre qui utilise une chaîne pour filtrer la requête Amazon Kendra.

```
"{"equalsTo": {"key": "City", "value": {"stringValue": "Seattle"}}}"
```

Voici un exemple de `queryFilterString` paramètre qui utilise un attribut de session appelé `"SourceURI"` pour filtrer la requête Amazon Kendra.

```
"{"equalsTo": {"key": "SourceURI","value": {"stringValue": "[FileURL]"}}}"
```

Voici un exemple de `queryFilterString` paramètre qui utilise un attribut de requête appelé `"DepartmentName"` pour filtrer la requête Amazon Kendra.

```
"{"equalsTo": {"key": "Department","value": {"stringValue": "((DepartmentName))"}}}"
```

Les `AMAZON.KendraSearchInteng` filtres utilisent le même format que les filtres de recherche Amazon Kendra. Pour plus d'informations, consultez la section Utilisation des attributs de document pour filtrer les résultats de recherche dans le guide du développeur Amazon Kendra.

La chaîne de filtre de requête utilisée avec le `AMAZON.KendraSearchIntent` doit utiliser des lettres minuscules pour la première lettre de chaque filtre. Par exemple, le filtre de requête suivant est valide pour le`AMAZON.KendraSearchIntent`.

```
{
"andAllFilters": [
    {
        "equalsTo": {
            "key": "City",
            "value": {
                "stringValue": "Seattle"
            }
        }
    },
    {
        "equalsTo": {
            "key": "State",
            "value": {
                "stringValue": "Washington"
            }
        }
    }
]
}
```

Utilisation de la réponse de recherche

Amazon Kendra renvoie la réponse à une recherche dans une réponse tirée de la déclaration d'`IntentClosingSetting`intention. L'intention doit comporter une `closingResponse` déclaration, sauf si une fonction Lambda produit un message de réponse de clôture.

Amazon Kendra propose cinq types de réponses.

- Les deux réponses suivantes nécessitent la configuration d'une FAQ pour votre index Amazon Kendra. Pour plus de détails, voir Ajouter des questions et réponses directement à un index.
  - `x-amz-lex:kendra-search-response-question_answer-question-<N>`— La question d'une FAQ qui correspond à la recherche.
  - `x-amz-lex:kendra-search-response-question_answer-answer-<N>`— La réponse d'une FAQ qui correspond à la recherche.
- Les trois réponses suivantes nécessitent la configuration d'une source de données pour votre index Amazon Kendra. Pour plus de détails, consultez la section Création d'une source de données.
  - `x-amz-lex:kendra-search-response-document-<N>`— Extrait d'un document de l'index lié au texte de l'énoncé.
  - `x-amz-lex:kendra-search-response-document-link-<N>`— L'URL d'un document dans l'index qui est lié au texte de l'énoncé.
  - `x-amz-lex:kendra-search-response-answer-<N>`— Un extrait d'un document de l'index qui répond à la question.

Les réponses sont renvoyées dans les attributs `request`. Il peut y avoir jusqu'à cinq réponses pour chaque attribut, numérotées de 1 à 5. Pour plus d'informations sur les réponses, consultez la section Types de réponses dans le manuel Amazon Kendra Developer Guide.

L'instruction `closingResponse` doit comporter un ou plusieurs groupes de messages. Chaque groupe de messages contient un ou plusieurs messages. Chaque message peut contenir une ou plusieurs variables d'espace réservé qui sont remplacées par des attributs de demande dans la réponse d'Amazon Kendra. Il doit y avoir au moins un message du groupe de messages dans lequel toutes les variables du message sont remplacées par des valeurs d'attribut de demande dans la réponse d'exécution, ou il doit y avoir un message du groupe de messages sans aucune variable d'espace réservé. Les attributs de demande sont définis avec des parenthèses doubles ("((" "))"). Les messages du groupe de messages suivants correspondent à toutes les réponses d'Amazon Kendra :

- « J'ai trouvé une question FAQ pour vous : ((x-amz-lex: kendra-search-response-question _answer-question-1)), et la réponse est ((x-amz-lex: _answer-answer-1)) » kendra-search-response-question
- « J'ai trouvé un extrait d'un document utile : ((x-amz-lex: kendra-search-response-document -1)) »
- « Je pense que la réponse à vos questions est ((x-amz-lex: kendra-search-response-answer -1)) »

Utilisation d'une fonction Lambda pour gérer la demande et la réponse

L'`AMAZON.KendraSearchIntent`intention peut utiliser votre crochet de code de dialogue et votre crochet de code d'expédition pour gérer la demande adressée à Amazon Kendra et la réponse. Utilisez la fonction Lambda du crochet de code de dialogue lorsque vous souhaitez modifier la requête que vous envoyez à Amazon Kendra, et la fonction Lambda du crochet de code d'expédition lorsque vous souhaitez modifier la réponse.

Création d'une requête avec le hook de code de dialogue

Vous pouvez utiliser le code hook de dialogue pour créer une requête à envoyer à Amazon Kendra. L'utilisation du hook de code de dialogue est facultative. Si vous ne spécifiez pas de crochet de dialogue, Amazon Lex V2 crée une requête à partir de l'énoncé de l'utilisateur et utilise `queryFilterString` celui que vous avez fourni lors de la configuration de l'intention, si vous en avez fourni un.

Vous pouvez utiliser deux champs dans la réponse au crochet de code de la boîte de dialogue pour modifier la demande adressée à Amazon Kendra :

- `kendraQueryFilterString`— Utilisez cette chaîne pour spécifier les filtres d'attributs pour la demande Amazon Kendra. Vous pouvez filtrer la requête à l'aide de l'un des champs d'index définis dans votre index. Pour connaître la structure de la chaîne de filtre, consultez la section Utilisation des attributs de document pour filtrer les requêtes dans le manuel Amazon Kendra Developer Guide. Si la chaîne de filtre spécifiée n'est pas valide, vous obtiendrez une exception `InvalidLambdaResponseException`. La chaîne `kendraQueryFilterString` remplace toute chaîne de requête spécifiée dans `queryFilterString` configuré pour l'intention.
- `kendraQueryRequestPayload`— Utilisez cette chaîne pour spécifier une requête Amazon Kendra. Votre requête peut utiliser n'importe laquelle des fonctionnalités d'Amazon Kendra. Si vous ne spécifiez pas de requête valide, vous obtenez une exception `InvalidLambdaResponseException`. Pour plus d'informations, consultez la section Requête dans le guide du développeur Amazon Kendra.

Après avoir créé le filtre ou la chaîne de requête, vous envoyez la réponse à Amazon Lex V2 avec le `dialogAction` champ de réponse défini sur`delegate`. Amazon Lex V2 envoie la requête à Amazon Kendra, puis renvoie la réponse à la requête au hook du code d'expédition.

Utilisation du hook de code d'exécution pour la réponse

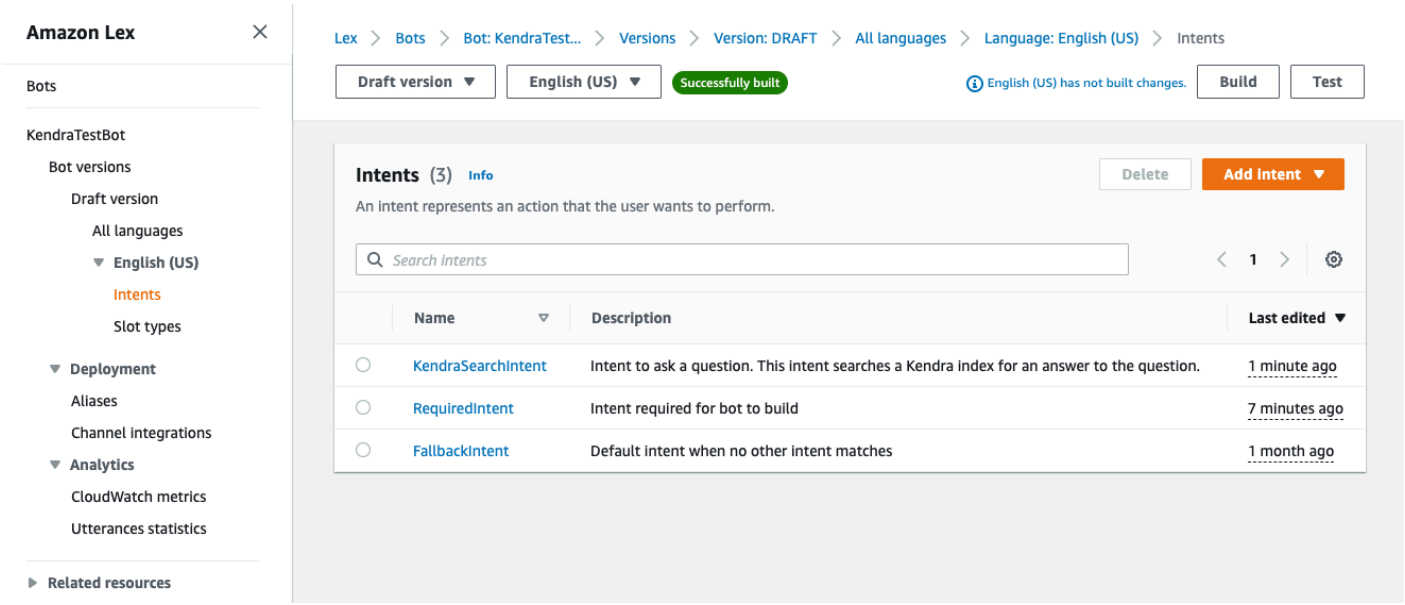
Une fois qu'Amazon Lex V2 a envoyé une requête à Amazon Kendra, la réponse à la requête est renvoyée à la fonction `AMAZON.KendraSearchIntent` Lambda d'expédition. L'événement d'entrée dans le code hook contient la réponse complète d'Amazon Kendra. Les données de requête ont la même structure que celles renvoyées par l'opération Amazon Kendra`Query`. Pour plus d'informations, consultez la section Syntaxe des réponses aux requêtes dans le manuel Amazon Kendra Developer Guide.

Le hook de code d'exécution est facultatif. S'il n'en existe pas, ou si le crochet de code ne renvoie aucun message dans la réponse, Amazon Lex V2 utilise l'`closingResponse`instruction pour les réponses.

Exemple : création d'un bot FAQ pour un index Amazon Kendra

Cet exemple crée un bot Amazon Lex V2 qui utilise un index Amazon Kendra pour fournir des réponses aux questions des utilisateurs. Le bot FAQ gère le dialogue pour l'utilisateur. Il utilise l'intention `AMAZON.KendraSearchIntent` pour interroger l'index et présenter la réponse à l'utilisateur. Voici un résumé de la façon dont vous allez créer votre bot FAQ à l'aide d'un index Amazon Kendra :

1. Créez un bot avec lequel vos clients interagiront pour obtenir des réponses.
2. Créez une intention personnalisée. `AMAZON.FallbackIntent`Les `AMAZON.KendraSearchIntent` et étant des intentions secondaires, votre bot a besoin d'au moins une autre intention qui doit contenir au moins un énoncé. Cette intention permet à votre bot de construire, mais n'est pas utilisée par ailleurs. Votre bot FAQ contiendra donc au moins trois intentions, comme dans l'image ci-dessous :



3. Ajoutez l'`AMAZON.KendraSearchIntent`intention à votre bot et configurez-le pour qu'il fonctionne avec votre index Amazon Kendra.
4. Testez le bot en effectuant une requête et en vérifiant que les résultats de votre index Amazon Kendra sont des documents qui répondent à la requête.

Prérequis

Avant de pouvoir utiliser cet exemple, vous devez créer un index Amazon Kendra. Pour plus d'informations, consultez Getting started with the Amazon Kendra console dans le guide du développeur Amazon Kendra. Pour cet exemple, choisissez l'exemple de jeu de données (exemple de documentation AWS) comme source de données.

Pour créer un bot FAQ, procédez comme suit :

1. Connectez-vous à la console Amazon Lex AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/lex/.
2. Dans le volet de navigation, sélectionnez Bots.
3. Choisissez Créer un bot.
   a. Pour la méthode de création, choisissez Create a blank bot.
   b. Dans la section Configuration du bot, attribuez au bot un nom indiquant son objectif, par exemple**`KendraTestBot`**, et une description facultative. Le nom doit être unique dans votre compte.

c. Dans la section Autorisations IAM, choisissez Créer un rôle avec des autorisations Amazon Lex de base. Cela créera un rôle AWS Identity and Access Management (IAM) doté des autorisations dont Amazon Lex V2 a besoin pour exécuter votre bot.

d. Dans la section Loi sur la protection de la vie privée en ligne des enfants (COPPA), choisissez Non.

e. Dans les sections Expiration de session inactive et Paramètres avancés, conservez les paramètres par défaut et choisissez Next.

f. Vous êtes maintenant dans la section Ajouter une langue au bot. Dans le menu sous Interaction vocale, sélectionnez Aucune. Il s'agit uniquement d'une application basée sur du texte. Conservez les paramètres par défaut pour les autres champs.

g. Sélectionnez Exécuté. Amazon Lex V2 crée votre bot et une intention par défaut est appelée NewIntent, puis vous dirige vers la page de configuration de cette intention

Pour créer un bot avec succès, vous devez créer au moins une intention distincte du `AMAZON.FallbackIntent` et du`AMAZON.KendraSearchIntent`. Cette intention est requise pour créer votre bot Amazon Lex V2, mais elle n'est pas utilisée pour la réponse à la FAQ. Cette intention doit contenir au moins un exemple d'énoncé et celui-ci ne doit s'appliquer à aucune des questions posées par votre client.

Pour créer l'intention requise :

1. Dans la section Détails de l'intention, donnez un nom à l'intention, tel que**`RequiredIntent`**.
2. Dans la section Exemples d'énoncés, tapez un énoncé dans la case située à côté de Ajouter un énoncé, tel que. **`Required utterance`** Choisissez ensuite Ajouter un énoncé.
3. Choisissez Save intent (Enregistrer l'intention).

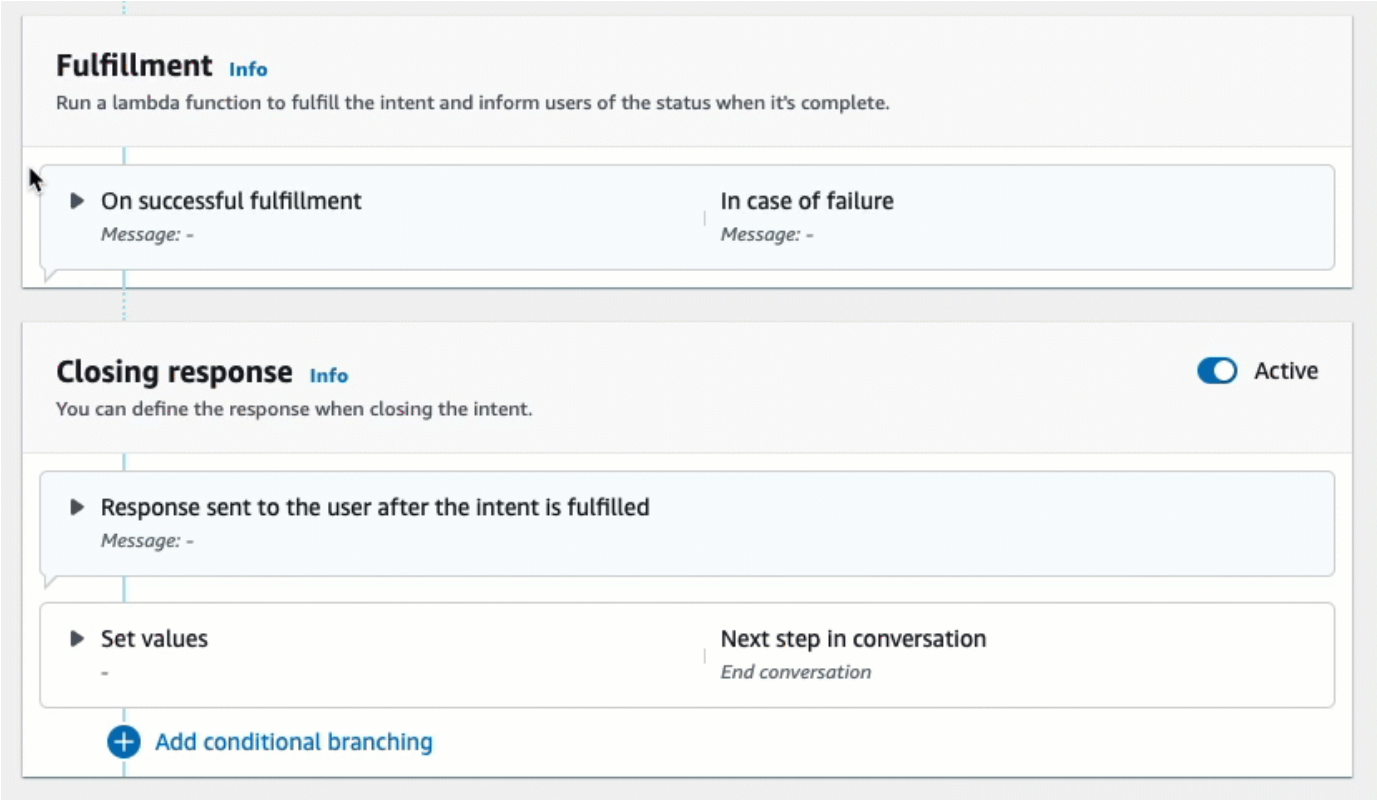
Créez l'intention de rechercher un index Amazon Kendra et le message de réponse qu'il doit renvoyer.

Pour créer un AMAZON. KendraSearchIntent intention et message de réponse :

1. Sélectionnez Retour à la liste des intentions dans le volet de navigation pour revenir à la page des intentions de votre bot. Choisissez Ajouter une intention, puis sélectionnez Utiliser une intention intégrée dans le menu déroulant.
2. Dans la zone qui apparaît, sélectionnez le menu sous Intention intégrée. Entrez **`AMAZON.KendraSearchIntent`** dans la barre de recherche, puis choisissez-la dans la liste.

3. Donnez un nom à l'intention, par exemple**`KendraSearchIntent`**.
4. Dans le menu déroulant de l'index Amazon Kendra, choisissez l'index que vous souhaitez rechercher. L'index que vous avez créé dans la section Prérequis doit être disponible.
5. Sélectionnez Ajouter.
6. Dans l'éditeur d'intention, faites défiler la page jusqu'à la section Expédition, sélectionnez la flèche droite pour développer la section et ajoutez le message suivant dans le champ Sous En cas de réussite de l'expédition :

```
I found a link to a document that could help you: ((x-amz-lex:kendra-search-
response-document-link-1)).
```



Pour plus d'informations sur la réponse de recherche Amazon Kendra, consultez la section Utilisation de la réponse de recherche.

7. Choisissez Save intent (Enregistrer l'intention), puis Build (Créer) pour créer le bot. Lorsque le bot est prêt, la bannière en haut de l'écran devient verte et affiche un message de réussite.

Enfin, utilisez la fenêtre de test de la console pour tester les réponses de votre bot.

Pour tester votre bot FAQ :

1. Une fois le bot créé avec succès, choisissez Test.
2. Entrez **`What is Amazon Kendra?`** dans la fenêtre de test de la console. Vérifiez que le bot répond par un lien.
3. Pour plus d'informations sur la configuration`AMAZON.KendraSearchIntent`, reportez-vous AMAZON.KendraSearchIntentaux sections et KendraConfiguration.

AMAZON.PauseIntent

Répond aux mots et aux phrases qui permettent à l'utilisateur de suspendre une interaction avec un bot afin de pouvoir y revenir plus tard. Votre fonction ou application Lambda doit enregistrer les données d'intention dans des variables de session, ou vous devez utiliser l'GetSessionopération pour récupérer les données d'intention lorsque vous reprenez l'intention actuelle.

Énoncés courants :

- pause
- mettez ça en pause

AMAZON.QnAIntent

> **Note**
>
> Avant de pouvoir tirer parti des fonctionnalités de l'IA générative, vous devez remplir les conditions préalables suivantes
>
> 1. Accédez à la console Amazon Bedrock et inscrivez-vous pour accéder au modèle Anthropic Claude que vous souhaitez utiliser (pour plus d'informations, voir Accès au modèle). Pour plus d'informations sur les tarifs d'utilisation d'Amazon Bedrock, consultez les tarifs d'Amazon Bedrock.
> 2. Activez les fonctionnalités d'IA générative pour les paramètres régionaux de votre bot. Pour ce faire, suivez les étapes indiquées surOptimisez la création et les performances des robots grâce à l'IA générative.

Répond aux questions des clients en utilisant un Amazon Bedrock FM pour rechercher et résumer les réponses aux FAQ. Cette intention est activée lorsqu'un énoncé n'est classé dans aucune des autres intentions présentes dans le bot. Notez que cette intention ne sera pas activée pour les énoncés manqués lors de l'obtention d'une valeur d'intervalle. Une fois reconnu, le`AMAZON.QnAIntent`, utilise le modèle Amazon Bedrock spécifié pour effectuer des recherches dans la base de connaissances configurée et répondre à la question du client.

Si la réponse du FM n'est pas satisfaisante ou si l'appel au FM échoue, Amazon Lex V2 invoque alors le. `AMAZON.FallbackIntent`

> **Warning**
>
> Vous ne pouvez pas utiliser le `AMAZON.QnAIntent` et `AMAZON.KendraSearchIntent` dans les mêmes paramètres régionaux de bot.

Les options de magasin de connaissances suivantes sont disponibles. Vous devez déjà avoir créé le magasin de connaissances et indexé les documents qu'il contient.

- OpenSearch Domaine de service : contient des documents indexés. Pour créer un domaine, suivez les étapes décrites dans Création et gestion des domaines Amazon OpenSearch Service.
- Index Amazon Kendra — Contient des documents de FAQ indexés. Pour créer un index Amazon Kendra, suivez les étapes décrites dans la section Création d'un index.
- Base de connaissances Amazon Bedrock : contient des sources de données indexées. Pour configurer une base de connaissances, suivez les étapes de la section Création d'une base de connaissances.

Si vous sélectionnez cette intention, vous configurez les champs suivants, puis sélectionnez Ajouter pour ajouter l'intention.

- Modèle Bedrock — Choisissez le fournisseur et le modèle de fondation à utiliser à cette fin. Actuellement, Anthropic Claude V2 et Anthropic Claude Instant sont pris en charge.
- Magasin de connaissances : choisissez la source à partir de laquelle vous souhaitez que le modèle extraie des informations pour répondre aux questions des clients. Les sources suivantes sont disponibles.
  - OpenSearch— Configurez les champs suivants.

- Point de terminaison de domaine : indiquez le point de terminaison de domaine que vous avez créé pour le domaine ou qui vous a été fourni après la création du domaine.
- Nom de l'index : indiquez l'index à rechercher. Pour plus d'informations, consultez Indexation des données dans Amazon OpenSearch Service.
- Choisissez la manière dont vous souhaitez renvoyer la réponse au client.
  - Réponse exacte — Lorsque cette option est activée, la valeur du champ Réponse est utilisée telle quelle pour la réponse du bot. Le modèle de base Amazon Bedrock configuré est utilisé pour sélectionner le contenu exact de la réponse tel quel, sans synthèse ni résumé du contenu. Spécifiez le nom des champs de question et de réponse configurés dans la OpenSearch base de données.
  - Inclure les champs : renvoie une réponse générée par le modèle à l'aide des champs que vous spécifiez. Spécifiez le nom d'un maximum de cinq champs configurés dans la OpenSearch base de données. Utilisez un point-virgule (;) pour séparer les champs.

- Amazon Kendra — Configurez les champs suivants.
  - Index Amazon Kendra : sélectionnez l'index Amazon Kendra dans lequel vous souhaitez que votre bot recherche.
  - Filtre Amazon Kendra : pour créer un filtre, cochez cette case. Pour plus d'informations sur le format JSON du filtre de recherche Amazon Kendra, consultez Utilisation des attributs de document pour filtrer les résultats de recherche.
  - Réponse exacte : pour que votre bot renvoie la réponse exacte renvoyée par Amazon Kendra, cochez cette case. Dans le cas contraire, le modèle Amazon Bedrock que vous sélectionnez génère une réponse basée sur les résultats.

  > **Note**
  >
  > Pour utiliser cette fonctionnalité, vous devez d'abord ajouter des questions de FAQ à votre index en suivant les étapes décrites dans la section Ajouter des questions fréquemment posées (FAQ) à un index.

- Base de connaissances Amazon Bedrock — Si vous choisissez cette option, spécifiez l'ID de la base de connaissances. Vous pouvez trouver l'ID en consultant la page de détails de la base de connaissances dans la console ou en envoyant une GetKnowledgeBasedemande.

Les réponses du QNAIntent seront stockées dans les attributs de la demande, comme indiqué ci-dessous :

- `x-amz-lex:qnA-search-response`— La réponse du QNAIntent à la question ou à l'énoncé.
- `x-amz-lex:qnA-search-response-source`— Pointe vers le document ou la liste de documents utilisés pour générer la réponse.

AMAZON.RepeatIntent

Répond aux mots et aux phrases qui permettent à l'utilisateur de répéter le message précédent. Votre application doit utiliser une fonction Lambda pour enregistrer les informations d'intention précédentes dans des variables de session, ou vous devez utiliser l'GetSessionopération pour obtenir les informations d'intention précédentes.

Énoncés courants :

- répéter
- redis-le
- Répète ça

AMAZON.ResumeIntent

Répond aux mots et aux phrases qui permettent à l'utilisateur de reprendre une intention précédemment interrompue. Votre fonction ou application Lambda doit gérer les informations requises pour reprendre l'intention précédente.

Énoncés courants :

- cv
- continuer
- continuez

AMAZON.StartOverIntent

Répond aux mots et aux phrases qui permettent à l'utilisateur d'arrêter de traiter l'intention actuelle et de recommencer depuis le début. Vous pouvez utiliser votre fonction Lambda ou l'`PutSession`opération pour obtenir à nouveau la première valeur d'emplacement.

Énoncés courants :

- recommencer
- redémarrer
- recommencer

AMAZON.StopIntent

Répond aux mots et aux phrases qui indiquent que l'utilisateur souhaite arrêter de traiter l'intention actuelle et mettre fin à l'interaction avec un bot. Votre fonction ou application Lambda doit effacer tous les attributs et valeurs de type d'emplacement existants, puis mettre fin à l'interaction.

Énoncés courants :

- stop
- off
- tais-toi

Ajouter des types de slots

Les types d'emplacements définissent les valeurs que les utilisateurs peuvent fournir pour vos variables d'intention. Vous définissez les types d'emplacements pour chaque langue afin que les valeurs soient spécifiques à cette langue. Par exemple, pour un type d'emplacement répertoriant les couleurs de peinture, vous pouvez inclure la valeur red « » en anglais, rouge « » en français et « rojo » en espagnol.

Cette rubrique explique comment créer des types de créneaux personnalisés qui fournissent des valeurs pour les créneaux de votre intention. Vous pouvez également utiliser les types d'emplacements intégrés pour les valeurs standard. Par exemple, vous pouvez utiliser le type de slot intégré `AMAZON.Country` pour une liste de pays dans le monde.

Pour créer un type de slot

1. Connectez-vous à la console Amazon Lex AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/lex/.
2. Dans la liste des robots, choisissez le bot auquel vous souhaitez ajouter la langue, puis choisissez Structure de conversation, puis Toutes les langues.
3. Choisissez la langue à laquelle ajouter le type de slot, puis choisissez Slot types.

4. Choisissez Ajouter un type d'emplacement, nommez votre type d'emplacement, puis choisissez Ajouter.
5. Dans l'éditeur de type d'emplacement, ajoutez les détails de votre type d'emplacement.

   - Résolution de la valeur des créneaux — Détermine la manière dont les valeurs des créneaux sont résolues. Si vous choisissez Expand values, Amazon Lex V2 utilise les valeurs comme valeurs représentatives pour l'entraînement. Si vous utilisez Restreindre aux valeurs d'emplacement, les valeurs autorisées pour l'emplacement sont limitées à celles que vous fournissez.
   - Valeurs du type d'emplacement : valeurs de l'emplacement. Si vous avez choisi Restreindre aux valeurs des emplacements, vous pouvez ajouter des synonymes à la valeur. Par exemple, pour la valeur « football », vous pouvez ajouter le synonyme « football ». Si l'utilisateur saisit « football » dans une conversation avec votre bot, la valeur réelle de la machine à sous est « football ».
   - Utiliser les valeurs des créneaux comme vocabulaire personnalisé : activez cette option pour améliorer la reconnaissance des valeurs des créneaux et des synonymes dans les conversations audio. N'activez pas cette option lorsque les valeurs des emplacements sont des termes courants, tels que « oui », « non », « un », « deux », « trois », etc.
6. Choisissez Enregistrer le type d'emplacement.

Amazon Lex V2 propose les types de machines à sous suivants :

Rubriques

Types de slots intégrés

Amazon Lex prend en charge les types d'emplacements intégrés qui définissent la manière dont les données de l'emplacement sont reconnues et traitées. Vous pouvez créer des options de ces types dans vos intentions. Cela vous évite ainsi de devoir créer des valeurs d'énumération pour les données d'option couramment utilisées, telles que la date, l'heure et l'emplacement. Les types d'options prédéfinies n'ont pas de versions.

Type d'option	Brève description	Paramètres régionaux pris en charge	
AMAZON. AlphaNumeric	Reconnaît les mots composés de lettres et de chiffres.	Toutes les langues sauf le coréen (Ko-KR)	
Amazon.City	Reconnaît les mots qui représentent une ville.	Toutes les localisations	
Amazon. Confirmation	Reconnaît les mots qui signifient « Oui », « Non », « Peut-être » et « Je ne sais pas » et les convertit dans un format standard (Oui/Non/Peut-être/Ne sais pas).	Anglais (en-US, en-GB, en-AU, en-IN, en-ZA)	
Amazon.country	Reconnaît les mots qui représentent un pays.	Toutes les localisations	
Amazon. Date	Reconnaît les mots qui représentent une date et les convertit dans un format standard.	Toutes les localisations	
Amazon. Durée	Reconnaît les mots qui représentent la durée et les convertit dans un format standard.	Toutes les localisations	

Type d'option	Brève description	Paramètres régionaux pris en charge	
AMAZON. EmailAddress	Reconnaît les mots qui représentent une adresse e-mail et les convertit en adresse e-mail standard.	Toutes les localisations	
AMAZON. FirstName	Reconnaît les mots qui représentent un prénom.	Toutes les localisations	
AMAZON. LastName	Reconnaît les mots qui représentent un nom de famille.	Toutes les localisations	
Numéro Amazon	Reconnaît les mots numériques et les convertit en chiffres.	Toutes les localisations	
AMAZON.Percentage	Reconnaît les mots qui représentent un pourcentage et les convertit en nombre et en signe de pourcentage (%).	Toutes les localisations	
AMAZON. PhoneNumber	Reconnaît les mots qui représentent un numéro de téléphone et les convertit en chaîne numérique.	Toutes les localisations	
État d'Amazon	Reconnaît les mots qui représentent un État.	Toutes les localisations	

Type d'option	Brève description	Paramètres régionaux pris en charge
AMAZON. StreetName	Reconnaît les mots qui représentent le nom d'une rue.	Toutes les localisations
Amazon Time	Reconnaît les mots qui indiquent l'heure et les convertit en format horaire.	Toutes les localisations
AMAZON.UKPostalCode	Reconnaît les mots qui représentent un code postal britannique et les convertit au format standard.	Anglais (britannique) (en-GB) uniquement
AMAZON. FreeFormInput	Reconnaît les chaînes composées de mots ou de caractères quelconques.	Toutes les localisations

AMAZON. AlphaNumeric

Reconnaît les chaînes composées de lettres et de chiffres, par exemple **APQ123**.

Ce type de machine à sous n'est pas disponible dans la région coréenne (Ko-KR).

Vous pouvez utiliser le type d'emplacement `AMAZON.AlphaNumeric` pour les chaînes qui contiennent :

- Des caractères alphabétiques, tels que **ABC**
- Des caractères numériques, tels que **123**
- Une combinaison de caractères alphanumériques, notamment **ABC123**

Le type de `AMAZON.AlphaNumeric` slot prend en charge les saisies utilisant des styles d'orthographe. Vous pouvez utiliser les spell-by-word styles spell-by-letter et pour aider vos clients à saisir des lettres. Pour de plus amples informations, veuillez consulter Capture de valeurs de créneaux avec des styles d'orthographe.

Vous pouvez ajouter une expression régulière au type d'emplacement `AMAZON.AlphaNumeric` pour valider les valeurs saisies pour l'emplacement. Par exemple, vous pouvez utiliser une expression régulière pour valider :

- Codes postaux canadiens
- Des numéros de permis de conduire
- Des numéros d'identification de véhicules

Utilisez une expression régulière standard. Amazon Lex V2 prend en charge les caractères suivants dans l'expression régulière :

- A-Z, a-z
- 0-9

Amazon Lex V2 prend également en charge les caractères Unicode dans les expressions régulières. Le formulaire est `\u`*`Unicode`*. Utilisez quatre chiffres pour représenter les caractères Unicode. Par exemple, `[\u0041-\u005A]` est équivalent à [A-Z].

Les opérateurs d'expression régulière suivants ne sont pas pris en charge :

- Répéteurs infinis :*, + ou {x,} sans limite supérieure.
- Caractère générique (.)

La longueur maximale de l'expression régulière est de 300 caractères. La longueur maximale d'une chaîne stockée dans un type de `AMAZON.AlphaNumeric` slot utilisant une expression régulière est de 30 caractères.

Voici quelques exemples d'expressions régulières.

- Des chaînes alphanumériques, telles que **`APQ123`** ou **`APQ1`** : `[A-Z]{3}[0-9]{1,3}` ou une plus limitée `[A-DP-T]{3} [1-5]{1,3}`
- Format US Postal Service Priority Mail International, tel que **`CP123456789US`** : `CP[0-9]{9}US`

- Numéros d'acheminement bancaire, tels que **123456789** : `[0-9]{9}`

Pour définir l'expression régulière d'un type d'emplacement, utilisez la console ou l'opération CreateSlotType. L'expression régulière est validée lorsque vous enregistrez le type d'emplacement. Si l'expression n'est pas valide, Amazon Lex V2 renvoie un message d'erreur.

Lorsque vous utilisez une expression régulière dans un type de slot, Amazon Lex V2 compare les entrées aux slots de ce type par rapport à l'expression régulière. Si l'entrée correspond à l'expression, la valeur est acceptée pour l'emplacement. Si l'entrée ne correspond pas, Amazon Lex V2 invite l'utilisateur à répéter la saisie.

Amazon.City

Fournit une liste des villes locales et mondiales. Le type de slot reconnaît les variantes courantes des noms de villes. Amazon Lex V2 ne convertit pas une variante en nom officiel.

Exemples :

- New York
- Reykjavik
- Tokyo
- Versailles

Amazon. Confirmation

Ce type d'emplacement reconnaît les phrases de saisie qui correspondent aux phrases et aux mots « Oui », « Non », « Peut-être » et « Je ne sais pas » pour Amazon Lex V2 et les convertit en l'une des quatre valeurs. Il peut être utilisé pour saisir la confirmation ou l'accusé de réception de l'utilisateur. Sur la base de la valeur finale résolue, vous pouvez créer des conditions pour concevoir plusieurs chemins de conversation.

Par exemple :

if {confirmation} = « Oui », répondez à l'intention

sinon, recherchez un autre emplacement

Exemples :

- Oui : Oui, oui, ok, bien sûr, je l'ai, je suis d'accord...
- Non : Non, négatif, non, oubliez ça, je vais refuser, pas question...
- Peut-être : C'est possible, peut-être, parfois, je pourrais, c'est peut-être vrai...
- Je ne sais pas : Je ne sais pas, Inconnu, Aucune idée, Je n'en suis pas sûr, Qui sait...

À compter du 17 août 2023, s'il existe un type d'emplacement personnalisé nommé « Confirmation », le nom doit être modifié pour éviter tout conflit avec la confirmation d'emplacement intégrée. Dans le menu de navigation de gauche de la console Lex, accédez au type d'emplacement (pour un type d'emplacement personnalisé existant nommé Confirmation) et mettez à jour le nom du type d'emplacement. Le nouveau nom du type d'emplacement ne doit pas être « Confirmation », qui est un mot clé réservé au type de créneau de confirmation intégré.

Amazon.country

Les noms des pays du monde entier. Exemples :

- Australie
- Allemagne
- Japon
- États-Unis
- Uruguay

Amazon. Date

Convertit les mots qui représentent des dates en un format de date.

La date est fournie à votre intention au format de date ISO-8601. La date à laquelle votre intention est reçue dans le slot peut varier en fonction de la phrase spécifique prononcée par l'utilisateur.

- Les énoncés qui correspondent à une date spécifique, tels que « aujourd'hui », « maintenant » ou « 25 novembre », sont convertis en une date complète :. `2020-11-25` Par défaut, il s'agit de dates identiques ou ultérieures à la date actuelle.
- Les énoncés renvoyant à une semaine future, tels que « semaine prochaine », sont convertis à la date du dernier jour de la semaine en cours. Au format ISO-8601, la semaine commence le lundi et se termine le dimanche. Par exemple, si aujourd'hui est le 25 novembre 2020, « semaine

prochaine » est converti en. `2020-11-29` Les dates correspondant à la semaine en cours ou à la semaine précédente sont converties au premier jour de la semaine. Par exemple, si aujourd'hui est le 25/11/2020, « la semaine dernière » est converti en. `2020-11-16`

- Les énoncés qui correspondent à un mois futur, mais pas à un jour précis, tels que « le mois suivant », sont convertis au dernier jour du mois. Par exemple, si aujourd'hui est le 25 novembre 2020, « mois suivant » est converti en. `2020-12-31` Pour les dates correspondant au mois en cours ou au mois précédent, convertissez-les au premier jour du mois. Par exemple, si aujourd'hui est le 25 novembre 2020, « ce mois-ci » correspond à. `2020-11-01`
- Les énoncés qui font référence à une année future, mais pas à un mois ou à un jour précis, tels que « l'année prochaine », sont convertis au dernier jour de l'année suivante. Par exemple, si aujourd'hui est le 25 novembre 2020, « l'année prochaine » est converti en. `2021-12-31` Pour les dates correspondant à l'année en cours ou à l'année précédente, convertissez-les au premier jour de l'année. Par exemple, si aujourd'hui est le 25/11/2020, « l'année dernière » est converti en. `2019-01-01`

Amazon. Durée

Convertit les mots qui indiquent des durées en durée numérique.

La durée est résolue dans un format basé sur le format de durée ISO-8601,. `PnYnMnWnDTnHnMnS` Le `P` indique qu'il s'agit d'une durée, d'une valeur numérique et que la lettre majuscule qui suit `n` est l'élément de date ou d'heure spécifique. `n` Par exemple, `P3D` cela signifie 3 jours. A `T` est utilisé pour indiquer que les valeurs restantes représentent des éléments temporels plutôt que des éléments de date.

Exemples :

- « dix minutes » : `PT10M`
- « cinq heures » : `PT5H`
- « trois jours » : `P3D`
- « quarante-cinq secondes » : `PT45S`
- « huit semaines » : `P8W`
- « sept ans » : `P7Y`
- « cinq heures dix minutes » : `PT5H10M`
- « deux ans trois heures dix minutes » : `P2YT3H10M`

AMAZON. EmailAddress

Reconnaît les mots qui représentent une adresse e-mail fournie comme nom d'utilisateur@domaine. Les adresses peuvent inclure les caractères spéciaux suivants dans un nom d'utilisateur : le trait de soulignement (_), le trait d'union (-), le point final (.) et le signe plus (+).

Le type de `AMAZON.EmailAddress` slot prend en charge les saisies utilisant des styles d'orthographe. Vous pouvez utiliser les spell-by-word styles spell-by-letter et pour aider vos clients à saisir des adresses e-mail. Pour de plus amples informations, veuillez consulter Capture de valeurs de créneaux avec des styles d'orthographe.

AMAZON. FirstName

Prénoms couramment utilisés. Ce type de machine à sous reconnaît les noms formels, les surnoms informels et les noms composés de plusieurs mots. Le nom envoyé à votre intention est la valeur envoyée par l'utilisateur. Amazon Lex V2 ne convertit pas le surnom en nom officiel.

Pour les prénoms qui se ressemblent mais qui sont orthographiés différemment, Amazon Lex V2 envoie à votre intention une forme commune unique.

Le type de `AMAZON.FirstName` slot prend en charge les saisies utilisant des styles d'orthographe. Vous pouvez utiliser les spell-by-word styles spell-by-letter et pour aider vos clients à saisir des noms. Pour de plus amples informations, veuillez consulter Capture de valeurs de créneaux avec des styles d'orthographe.

Exemples :

- Emilie
- John
- Sophie
- Anil Kumar

AMAZON. FirstName renvoie également une liste de noms étroitement liés en fonction de la valeur d'origine. Vous pouvez utiliser la liste des valeurs résolues pour corriger des fautes de frappe, confirmer le nom auprès de l'utilisateur ou effectuer une recherche dans la base de données pour trouver des noms valides dans votre annuaire des utilisateurs.

Par exemple, l'entrée « John » peut entraîner le renvoi de noms connexes supplémentaires tels que « John J » et « John-Paul ».

Voici le format de réponse pour le type de slot `AMAZON.FirstName` intégré :

```
"value": {
    "originalValue": "John",
    "interpretedValue": "John",
    "resolvedValues": [
        "John",
        "John J.",
        "John-Paul"
    ]
}
```

AMAZON. LastName

Noms de famille couramment utilisés. Pour les noms qui se ressemblent et qui sont orthographiés différemment, Amazon Lex V2 envoie à votre intention une forme commune unique.

Le type de `AMAZON.LastName` slot prend en charge les saisies utilisant des styles d'orthographe. Vous pouvez utiliser les spell-by-word styles spell-by-letter et pour aider vos clients à saisir des noms. Pour de plus amples informations, veuillez consulter Capture de valeurs de créneaux avec des styles d'orthographe.

Exemples :

- Brosky
- Dasher
- Evers
- Parres
- Monde

AMAZON. LastName renvoie également une liste de noms étroitement liés en fonction de la valeur d'origine. Vous pouvez utiliser la liste des valeurs résolues pour corriger des fautes de frappe, confirmer le nom auprès de l'utilisateur ou effectuer une recherche dans la base de données pour trouver des noms valides dans votre annuaire des utilisateurs.

Par exemple, l'entrée « Smith » peut entraîner le renvoi de noms connexes supplémentaires tels que « Smyth » et « Smithe ».

Voici le format de réponse pour le type de slot `AMAZON.LastName` intégré :

```
"value": {
    "originalValue": "Smith",
    "interpretedValue": "Smith",
    "resolvedValues": [
        "Smith",
        "Smyth",
        "Smithe"
    ]
}
```

Numéro Amazon

Convertit les mots ou les nombres qui expriment un nombre en chiffres, y compris en nombres décimaux. Le tableau suivant montre la façon dont le type d'option `AMAZON.Number` capture les mots numériques.

Entrée	Réponse
cent vingt-trois point quatre cinq	123.45
cent vingt-trois point quarante cinq	123.45
zéro point quatre deux	0.42
zéro point quarante deux	0.42
232.998	232.998
50	50
-15	-15
moins 15	-15

AMAZON.Percentage

Convertit les mots et les symboles qui représentent un pourcentage en valeur numérique accompagné du signe de pourcentage (%).

Si l'utilisateur entre un nombre sans signe de pourcentage ou sans le mot « pour cent », la valeur de l'option sera un nombre. Le tableau suivant montre la façon dont le type d'option `AMAZON.Percentage` capture les pourcentages.

Entrée	Réponse
50 pour cent	50%
0,4 pour cent	0.4%
23,5 %	23,5%
vingt-cinq pour cent	25%

AMAZON. PhoneNumber

Convertit les nombres ou mots qui représentent un numéro de téléphone en format de chaîne sans ponctuation, comme suit.

Type	Description	Entrée	Résultat
Numéro international avec le signe plus (+) au début	Numéro à 11 chiffres avec le signe plus (+) au début	+61 7 4445 1061 +1 (509) 555-1212	`+61744431061` `+15095551212`
Numéro international sans signe plus (+)	Numéro à 11 chiffres sans signe plus	1 (509) 555-1212 61 7 4445 1061	`15095551212` `61744451061`
Numéro national	Numéro à 10 chiffres sans code internati onal	(03) 5115 4444 (509) 555-1212	`0351154444` `5095551212`
Numéro local	numéro de téléphone sans code internati onal ou indicatif régional	555-1212	`5551212`

État d'Amazon

Les noms des régions géographiques et politiques des pays.

Exemples :

- Bavière
- Préfecture de Fukushima
- Nord-Ouest du Pacifique
- Queensland
- Pays de Galles

AMAZON. StreetName

Les noms des rues comprises dans une adresse postale typique. Cela inclut uniquement le nom de la rue, pas le numéro de la maison.

Exemples :

- Avenue de Canberra
- Front Street
- Route du marché

Amazon Time

Convertit les mots qui représentent des heures en valeurs temporelles. `AMAZON.Time`peut résoudre les heures exactes, les valeurs ambiguës et les plages de temps. La valeur du slot peut être résolue selon les plages de temps suivantes :

- AM
- PM
- MO (matin)
- AF (après-midi)
- EV (soirée)
- NI (nuit)

Lorsqu'un utilisateur saisit une heure ambiguë, Amazon Lex V2 utilise l'`slots`attribut d'un événement Lambda pour transmettre les résolutions relatives aux heures ambiguës à votre fonction Lambda. Par exemple, si le bot demande à l'utilisateur une heure de livraison, l'utilisateur peut répondre « 10 heures ». Cette heure est ambiguë. Elle peut aussi bien signifier 10 h du matin que 10 h du soir. Dans ce cas, la valeur du `interpretedValue` champ est`null`, et le `resolvedValues` champ contient les deux résolutions possibles de l'heure. Amazon Lex V2 saisit les informations suivantes dans la fonction Lambda :

```
"slots": {
    "deliveryTime": {
    "value": {
        "originalValue": "10 o'clock",
        "interpretedValue": null,
        "resolvedValues": [
            "10:00", "22:00"
        ]
    }
}
```

Lorsque l'utilisateur répond en indiquant une heure sans ambiguïté, Amazon Lex V2 envoie l'heure à votre fonction Lambda dans `interpretedValue` le champ de l'attribut de `slots` l'événement Lambda. Par exemple, si votre utilisateur répond à l'invite indiquant l'heure de livraison par « 10 h 00 », Amazon Lex V2 saisit ce qui suit dans la fonction Lambda :

```
"slots": {
    "deliveryTime": {
        "value": {
        "originalValue": "10 AM",
        "interpretedValue": 10:00,
        "resolvedValues": [
            "10:00"
        ]
    }
}
```

Lorsque l'utilisateur répond à une demande d'heure de livraison par « le matin », Amazon Lex V2 saisit ce qui suit dans la fonction Lambda :

```
"slots": {
    "deliveryTime": {
```

```
    "value": {
        "originalValue": "morning",
        "interpretedValue": "MO",
        "resolvedValues": [
            "MO"
        ]
    }
}
```

Pour plus d'informations sur les données envoyées par Amazon Lex V2 à une fonction Lambda, consultez. Interprétation du format d'événement d'entrée

AMAZON.UK PostalCode

Convertit les mots représentant un code postal britannique en un format standard pour les codes postaux du Royaume-Uni. Le type de `AMAZON.UKPostalCode` slot valide et résout le code postal dans un ensemble de formats standardisés, mais il ne vérifie pas la validité du code postal. Votre candidature doit valider le code postal.

Le type de `AMAZON.UKPostalCode` slot n'est disponible que dans la langue anglaise (Royaume-Uni) (en-GB).

Le type de `AMAZON.UKPostalCode` slot prend en charge les saisies utilisant des styles d'orthographe. Vous pouvez utiliser les spell-by-word styles spell-by-letter et pour aider vos clients à saisir des lettres. Pour de plus amples informations, veuillez consulter Capture de valeurs de créneaux avec des styles d'orthographe.

Le type de slot reconnaît uniquement les formats de code postal valides listés ci-dessous, utilisés au Royaume-Uni. Les formats valides sont les suivants (« A » représente une lettre et « 9 » représente un chiffre) :

- AA9A 9AA
- A9A 9AA
- A9 9AA
- A99 9AA
- AA9 9AA
- AA99 9AA

Pour la saisie de texte, l'utilisateur peut saisir n'importe quelle combinaison de lettres majuscules et minuscules. L'utilisateur peut utiliser ou omettre l'espace dans le code postal. La valeur résolue inclura toujours l'espace à l'endroit approprié pour le code postal.

Pour la saisie vocale, l'utilisateur peut prononcer les caractères individuels ou utiliser des prononciations de lettres doubles, telles que « double A » ou « double 9 ». Ils peuvent également utiliser des prononciations à deux chiffres, comme « quatre-vingt-dix-neuf » pour « 99 ».

> **Note**
>
> Les codes postaux britanniques ne sont pas tous reconnus. Seuls les formats listés ci-dessus sont pris en charge.

AMAZON. FreeFormInput

`AMAZON.FreeFormInput`peut être utilisé pour capturer les entrées sous forme libre de l'utilisateur final. Il reconnaît les chaînes composées de mots ou de caractères. La valeur résolue est l'intégralité de l'énoncé d'entrée.

Exemple :

Robot : veuillez nous faire part de vos commentaires sur votre expérience d'appel.

Utilisateur : J'ai obtenu les réponses à toutes mes questions et j'ai pu terminer la transaction.

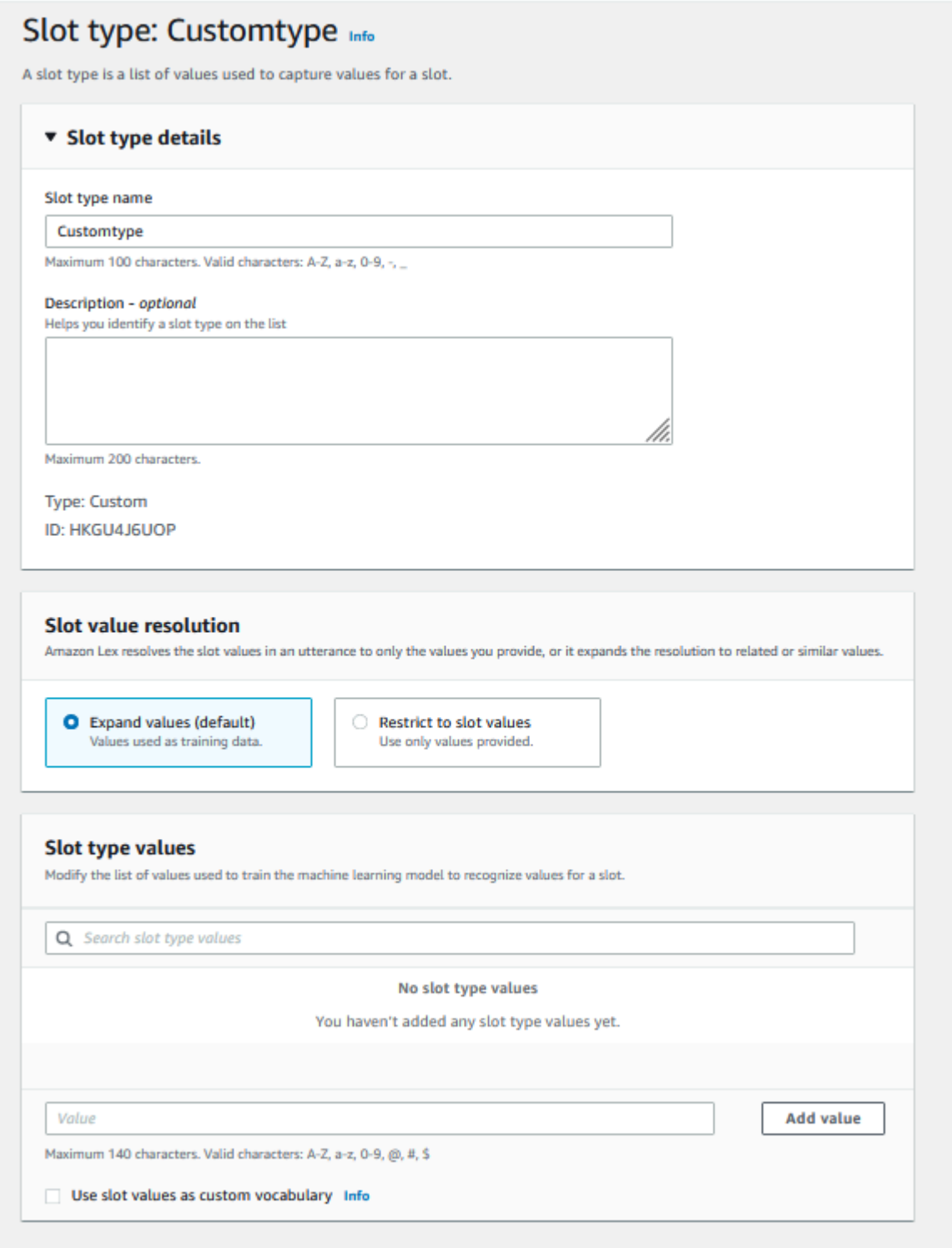
Remarque :

- `AMAZON.FreeFormInput`peut être utilisé pour capturer la saisie sous forme libre telle quelle par l'utilisateur final.
- `AMAZON.FreeFormInput`ne peut pas être utilisé dans des exemples d'énoncés d'intention.
- `AMAZON.FreeFormInput`ne peut pas contenir d'extraits d'énoncés.
- `AMAZON.FreeFormInput`n'est reconnu que lorsqu'il est sollicité.
- `AMAZON.FreeFormInput`ne prend pas en charge le mode Wait and Continue.
- `AMAZON.FreeFormInput`n'est actuellement pas pris en charge sur le canal Amazon Connect Chat.
- Lorsqu'un `AMAZON.FreeFormInput` emplacement est obtenu, il ne `FallbackIntent` sera pas déclenché.

- Lorsqu'un `AMAZON.FreeFormInput` emplacement est obtenu, il n'y aura aucun changement d'intention.

Type de slot personnalisé

Pour chaque intention, vous pouvez spécifier des paramètres qui indiquent les informations dont l'intention a besoin pour traiter la demande de l'utilisateur. Ces paramètres, ou options, sont associés à un type. Un type d'emplacement est une liste de valeurs qu'Amazon Lex V2 utilise pour entraîner le modèle d'apprentissage automatique à reconnaître les valeurs d'un emplacement. Par exemple, vous pouvez définir un type de machine à sous appelé `Genres` avec des valeurs telles que « comédie », « aventure », « documentaire », etc. Vous pouvez définir des synonymes pour une valeur de type de slot. Par exemple, vous pouvez définir les synonymes « funny » et « humorous » pour la valeur « comedy ».



Vous pouvez configurer le type d'emplacement pour augmenter les valeurs des emplacements. Les valeurs des créneaux seront utilisées comme données d'apprentissage et le modèle résoudra le créneau à la valeur fournie par l'utilisateur si elle est similaire aux valeurs des créneaux et des synonymes de ces valeurs. Il s'agit du comportement de par défaut. Amazon Lex V2 tient à jour une

liste des résolutions possibles pour un emplacement. Chaque entrée de la liste fournit une valeur résolue qu'Amazon Lex V2 a reconnue comme des possibilités supplémentaires pour le slot. Une valeur résolue est le meilleur effort pour correspondre à la valeur de l'emplacement. La liste peut contenir jusqu'à cinq valeurs.

Vous pouvez également configurer le type d'emplacement pour limiter la résolution aux valeurs des emplacements. Dans ce cas, le modèle résoudra une valeur d'emplacement saisie par l'utilisateur en une valeur d'emplacement existante uniquement si elle est identique à cette valeur d'emplacement ou s'il s'agit d'un synonyme. Par exemple, si l'utilisateur entre « funny », son entrée sera associée à la valeur d'option « comedy ».

Lorsque la valeur saisie par l'utilisateur est synonyme d'une valeur de type de slot, le modèle renvoie cette valeur de type de slot comme première entrée de la liste des`resolvedValues`. Par exemple, si l'utilisateur saisit « drôle », le modèle remplit le `originalValue` champ avec la valeur « drôle » et la première entrée du champ ResolvedValues avec « comédie ». Vous pouvez configurer `valueSelectionStrategy` lorsque vous créez ou mettez à jour un type d'option avec l'opération CreateSlotType de manière à ce que la valeur d'option inclut la première valeur de la liste de la résolution.

Les types d'emplacements personnalisés prennent en charge les entrées utilisant des styles d'orthographe. Vous pouvez utiliser les spell-by-word styles spell-by-letter et pour aider vos clients à saisir des lettres. Pour de plus amples informations, veuillez consulter Capture de valeurs de créneaux avec des styles d'orthographe.

Si vous utilisez une fonction Lambda, l'événement d'entrée de la fonction inclut une liste de résolution appelée. `resolvedValues` L'exemple suivant montre la section slot de l'entrée d'une fonction Lambda :

```
"slots": {
   "MovieGenre": {
      "value": {
         "originalValue": "funny",
         "interpretedValue": "comedy",
         "resolvedValues": [
            "comedy"
         ]
      }
   }
}
```

Chaque type d'option peut inclure jusqu'à 10 000 valeurs et synonymes. Chaque bot peut inclure jusqu'à 50 000 valeurs de types d'options et synonymes. Par exemple, vous pouvez avoir 5 types d'option, chacun avec 5 000 valeurs et 5 000 synonymes, ou vous pouvez avoir 10 types d'options, chacun avec 2 500 valeurs et 2 500 synonymes.

Un type d'emplacement personnalisé ne doit pas porter le même nom que les types d'emplacement intégrés. Par exemple, un type de créneau personnalisé ne doit pas être nommé avec les mots clés réservés Date, Numéro ou Confirmation. Ces mots clés sont réservés aux types d'emplacements intégrés. Pour obtenir la liste de tous les types d'emplacements intégrés, consultezTypes de slots intégrés.

Type d'emplacement de grammaire

Avec le type d'emplacement de grammaire, vous pouvez créer votre propre grammaire au format XML conformément à la spécification SRGS afin de collecter des informations dans une conversation. Amazon Lex V2 reconnaît les énoncés correspondant aux règles spécifiées dans la grammaire. Vous pouvez également fournir des règles d'interprétation sémantique à l'aide de balises ECMAScript dans les fichiers de grammaire. Amazon Lex renvoie ensuite les propriétés définies dans les balises sous forme de valeurs résolues lorsqu'une correspondance se produit.

Vous ne pouvez créer des types de cases grammaticales que dans les langues anglaise (Australie), anglaise (Royaume-Uni) et anglaise (États-Unis).

Un type d'emplacement grammatical comporte deux parties. Le premier est la grammaire elle-même écrite à l'aide du format de spécification SRGS. La grammaire interprète l'énoncé de l'utilisateur. Si l'énoncé est accepté par la grammaire, il correspond, sinon il est rejeté. Si un énoncé correspond, il est transmis au script s'il en existe un.

Le second, qui fait partie d'un type de slot grammatical, est un script optionnel écrit en ECMAScript qui transforme l'entrée en valeurs résolues renvoyées par le type de slot. Par exemple, vous pouvez utiliser un script pour convertir des nombres prononcés en chiffres. <tag>Les instructions ECMAScript sont incluses dans l'élément.

L'exemple suivant est au format XML conformément à la spécification SRGS qui montre une grammaire valide acceptée par Amazon Lex V2. Il définit un type d'emplacement grammatical qui accepte les numéros de cartes et détermine s'ils concernent des comptes réguliers ou premium. Pour plus d'informations sur la syntaxe acceptable, consultez les Format de script rubriques Définition grammaticale et.

```
<grammar version="1.0" xmlns="http://www.w3.org/2001/06/grammar"
```

```
         xml:lang="en-US" tag-format="semantics/1.0" root="card_number">

   <rule id="card_number" scope="public">
        <item repeat="0-1">
            card number
        </item>
        <item>
            seven
            <tag>out.value = "7";</tag>
         </item>
         <item>
            <one-of>
               <item>
                  two four one
                  <tag> out.value = out.value + "241"; out.card_type = "premium"; </
tag>
               </item>
               <item>
                  zero zero one
                  <tag> out.value = out.value + "001"; out.card_type = "regular";</tag>
               </item>
            </one-of>
         </item>
   </rule>
</grammar>
```

La grammaire ci-dessus n'accepte que deux types de numéros de cartes : 7241 ou 7001. Ces deux éléments peuvent éventuellement être préfixés par « numéro de carte ». Il contient également des balises ECMAScript qui peuvent être utilisées pour l'interprétation sémantique. Avec une interprétation sémantique, l'énoncé « carte numéro sept deux quatre un » renverrait l'objet suivant :

```
{
    "value": "7241",
    "card_type": "premium"
}
```

Cet objet est renvoyé sous forme de chaîne sérialisée JSON dans l'`resolvedValues`objet renvoyé par les opérations RecognizeTextRecognizeUtterance, et. StartConversation

Ajouter un type d'emplacement grammatical

Pour ajouter un type d'emplacement grammatical

1. Téléchargez la définition XML de votre type d'emplacement dans un compartiment S3. Notez le nom du bucket et le chemin d'accès au fichier.

   > **Note**
   >
   > La taille maximale du fichier est de 100 Ko.

2. Connectez-vous à la console Amazon Lex AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/lex/.
3. Dans le menu de gauche, choisissez Bots, puis choisissez le bot auquel ajouter le type d'emplacement grammatical.
4. Choisissez Afficher les langues, puis choisissez la langue à laquelle vous souhaitez ajouter le type d'emplacement grammatical.
5. Choisissez Afficher les types de créneaux.
6. Choisissez Ajouter un type de slot, puis choisissez Ajouter un type de slot grammatical.
7. Donnez un nom au type d'emplacement, puis choisissez Ajouter.
8. Choisissez le compartiment S3 qui contient votre fichier de définition et entrez le chemin d'accès au fichier. Choisissez Enregistrer le type d'emplacement.

Définition grammaticale

Cette rubrique présente les parties de la spécification SRGS prises en charge par Amazon Lex V2. Toutes les règles sont définies dans la spécification SRGS. Pour plus d'informations, consultez la recommandation W3C de la version 1.0 de la spécification grammaticale de reconnaissance vocale.

Rubriques

- Déclarations d'en-tête
- Éléments XML pris en charge
- Jetons
- Référence de règle
- Séquences et encapsulation

- Répète
- Langage
- Étiquettes
- Poids

Ce document inclut du matériel copié et dérivé de la version 1.0 de la spécification grammaticale de reconnaissance vocale du W3C (disponible sur https://www.w3.org/TR/speech-grammar/). Les informations de citation sont les suivantes :

Le document de spécification SRGS, une recommandation du W3C, est disponible auprès du W3C sous la licence suivante.

Texte de licence

Licence

En utilisant et/ou en copiant ce document, ou le document du W3C auquel cette déclaration est liée, vous (le licencié) confirmez que vous avez lu, compris et que vous vous engagez à respecter les termes et conditions suivants :

L'autorisation de copier et de distribuer le contenu de ce document, ou du document du W3C auquel cette déclaration est liée, sur n'importe quel support, à quelque fin que ce soit et sans frais ni redevance est accordée par la présente, à condition que vous incluiez les informations suivantes sur TOUTES les copies du document, ou parties de celui-ci, que vous utilisez :

- Un lien ou une URL vers le document original du W3C.
- La notice de copyright préexistante de l'auteur original ou, si elle n'existe pas, une notice (l'hypertexte est préférable, mais une représentation textuelle est autorisée) de la forme : « Copyright © [$date-of-document] World Wide Web Consortium, (MIT, ERCIM, Keio, Beihang). http://www.w3.org/Consortium/Legal/2015/doc-license »
- S'il existe, le STATUT du document du W3C.

Lorsque l'espace le permet, le texte intégral de cet AVIS doit être inclus. Nous demandons que l'attribution de la paternité soit fournie dans tout logiciel, document ou autre élément ou produit que

vous créez conformément à la mise en œuvre du contenu de ce document, ou de toute partie de celui-ci.

Aucun droit de créer des modifications ou des dérivés des documents du W3C n'est accordé en vertu de cette licence, sauf dans les cas suivants : pour faciliter la mise en œuvre des spécifications techniques énoncées dans ce document, n'importe qui peut préparer et distribuer des œuvres dérivées et des parties de ce document dans un logiciel, dans des supports accompagnant le logiciel et dans la documentation du logiciel, À CONDITION que tous ces travaux incluent la notice ci-dessous. CEPENDANT, la publication d'œuvres dérivées de ce document à des fins de spécification technique est expressément interdite.

En outre, les « composants du code » (Web IDL dans les sections clairement identifiées comme Web IDL ; le balisage défini par le W3C (HTML, CSS, etc.) et le code du langage de programmation informatique clairement indiqué comme exemples de code — sont sous licence logicielle du W3C.

L'avis est le suivant :

« Droits d'auteur © 2015 W3C® (MIT, ERCIM, Keio, Beihang). Ce logiciel ou ce document inclut du matériel copié ou dérivé de [titre et URI du document du W3C]. »

Avertissements

CE DOCUMENT EST FOURNI « TEL QUEL » ET LES DÉTENTEURS DES DROITS D'AUTEUR NE FONT AUCUNE DÉCLARATION NI NE DONNENT AUCUNE GARANTIE, EXPRESSE OU IMPLICITE, Y COMPRIS, MAIS SANS S'Y LIMITER, LES GARANTIES DE QUALITÉ MARCHANDE, D'ADÉQUATION À UN USAGE PARTICULIER, D'ABSENCE DE CONTREFAÇON OU DE TITRE ; QUE LE CONTENU DU DOCUMENT CONVIENT À QUELQUE USAGE QUE CE SOIT ; NI QUE LA MISE EN ŒUVRE DE CE CONTENU NE VIOLERA AUCUN BREVET, DROIT D'AUTEUR, MARQUE COMMERCIALE OU AUTRE DROIT TIERS.

LES DÉTENTEURS DES DROITS D'AUTEUR NE SERONT PAS RESPONSABLES DES DOMMAGES DIRECTS, INDIRECTS, SPÉCIAUX OU CONSÉCUTIFS RÉSULTANT DE L'UTILISATION DU DOCUMENT OU DE L'EXÉCUTION OU DE LA MISE EN ŒUVRE DE SON CONTENU.

Le nom et les marques commerciales des détenteurs des droits d'auteur ne peuvent PAS être utilisés dans des publicités relatives à ce document ou à son contenu sans autorisation écrite spécifique. Les droits d'auteur de ce document resteront à tout moment la propriété des détenteurs des droits d'auteur.

Déclarations d'en-tête

Le tableau suivant présente les déclarations d'en-tête prises en charge par le type d'emplacement grammatical. Pour plus d'informations, consultez les déclarations d'en-tête grammaticales dans la recommandation W3C de la version 1 de la spécification grammaticale de reconnaissance vocale.

Déclaration	Exigence de spécification	formulaire XML	Assistance Amazon Lex	Spécification
Version grammaticale	Obligatoire	4.3 : `version` attribut sur `grammar` un élément	Obligatoire	SRGS
Espace de noms XML	Obligatoire (XML uniquement)	4.3 : `xmlns` attribut sur `grammar` un élément	Obligatoire	SRGS
Type de document	Obligatoire (XML uniquement)	4.3 : DOCTYPE XML	Recommandée	SRGS
Encodage de caractères	Recommandée	4.4 : `encoding` attribut dans la déclaration XML	Recommandée	SRGS
Langage	Obligatoire en mode vocal Ignoré en mode DTMF	4.5 : `xml:lang` attribut sur `grammar` un élément	Obligatoire en mode vocal Ignoré en mode DTMF	SRGS
Mode	Facultatif	4.6 : `mode` attribut sur `grammar` un élément	Facultatif	SRGS
Règle racine	Facultatif	4.7 : `root` attribut sur	Obligatoire	SRGS

Déclaration	Exigence de spécification	formulaire XML	Assistance Amazon Lex	Spécification
		`grammar` un élément		
Format de balise	Facultatif	4.8 : `tag-format` attribut sur l'`grammar`élément	La chaîne, le littéral et l'ECMAScript sont pris en charge.	SERGS, SŒUR
URI de base	Facultatif	4.9 : `xml:base` attribut sur `grammar` un élément	Facultatif	SRGS
Lexique de prononciation	Facultatif, plusieurs options autorisées	4.10 : élément `lexicon`	Non pris en charge	SRGS, SVP
Metadonnées	Facultatif, plusieurs options autorisées	4.11.1 : élément `meta`	Obligatoire	SRGS
métadonnées XML	Facultatif, XML uniquement	4.11.2 : élément `metadata`	Facultatif	SRGS
Tag	Facultatif, plusieurs options autorisées	4.12 : élément `tag`	Les balises globales ne sont pas prises en charge	SRGS

Example (Exemple)

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE grammar PUBLIC "-//W3C//DTD GRAMMAR 1.0//EN"
                 "http://www.w3.org/TR/speech-grammar/grammar.dtd">
```

```
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xml:base="http://www.example.com/base-file-path"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US"
         version="1.0"
         mode="voice"
         root="city"
         tag-format="semantics/1.0">
```

Éléments XML pris en charge

Amazon Lex V2 prend en charge les éléments XML suivants pour les grammaires personnalisées :

- `<item>`
- `<token>`
- `<tag>`
- `<one-of>`
- `<rule-ref>`

Jetons

Le tableau suivant présente les spécifications des jetons prises en charge par le type d'emplacement grammatical. Pour plus d'informations, consultez la section Jetons dans la recommandation du W3C de la version 1 de la spécification grammaticale de reconnaissance vocale.

Type de jeton	Exemple	Pris en charge ?
Jeton unique sans guillemets	bonjour	Oui
Jeton unique sans guillemets : non alphabétique	2	Oui
Jeton entre guillemets simples, pas d'espace blanc	"hello"	Oui, supprimez les guillemet s doubles lorsqu'il ne contient qu'un seul jeton

Type de jeton	Exemple	Pris en charge ?
Deux jetons délimités par un espace blanc	bon voyage	Oui
Quatre jetons délimités par un espace blanc	c'est un test	Oui
Jeton entre guillemets simples, espaces blancs inclus	« San Francisco	Non
Jeton XML unique dans la <token>balise	<token>San Francisco</token>	Non (identique à un jeton entre guillemets simples avec espace blanc)

Remarques

- Jeton entre guillemets simples, espaces blancs inclus — La spécification exige que les mots placés entre guillemets doubles soient traités comme un jeton simple. Amazon Lex V2 les traite comme des jetons délimités par des espaces blancs.
- Entrée d'un jeton XML unique <token>: la spécification nécessite des mots délimités par <token>pour représenter un jeton. Amazon Lex V2 les traite comme des jetons délimités par des espaces blancs.
- Amazon Lex V2 génère une erreur de validation lorsque l'une des utilisations est détectée dans votre grammaire.

Example (Exemple)

```
<rule id="state" scope="public">
    <one-of>
        <item>FL</item>
        <item>MA</item>
        <item>NY</item>
    </one-of>
</rule>
```

Référence de règle

Le tableau suivant résume les différentes formes de référence aux règles qui sont possibles dans les documents de grammaire. Pour plus d'informations, voir la référence aux règles dans la recommandation W3C de la version 1 de la spécification grammaticale de reconnaissance vocale.

Type de référence	formulaire XML	Pris en charge
2.2.1 Référence explicite aux règles locales	`<ruleref uri="#rulename"/>`	Oui
2.2.2 Référence explicite à une règle nommée d'une grammaire identifiée par un URI	`<ruleref uri="grammarURI#rulename"/>`	Non
2.2.2 Référence implicite à la règle racine d'une grammaire identifiée par un URI	`<ruleref uri="grammarURI"/>`	Non
2.2.2 Référence explicite à une règle nommée d'une grammaire identifiée par un URI avec un type de média	`<ruleref uri="grammarURI#rulename" type="media-type"/>`	Non
2.2.2 Référence implicite à la règle racine d'une grammaire identifiée par un URI avec un type de média	`<ruleref uri="grammarURI" type="media-type"/>`	Non

2.2.3 Définitions des règles spéciales	`<ruleref special="NULL"/>`
	`<ruleref special="VOID"/>`
	`<ruleref special="GARBAGE"/>`
		Non

Remarques

1. L'URI de grammaire est un URI externe. Par exemple, `http://grammar.example.com/world-cities.grxml`.
2. Le type de support peut être :
   - `application/srgs+xml`
   - `text/plain`

Example (Exemple)

```
<rule id="city" scope="public">
    <one-of>
        <item>Boston</item>
        <item>Philadelphia</item>
        <item>Fargo</item>
    </one-of>
</rule>

<rule id="state" scope="public">
    <one-of>
        <item>FL</item>
        <item>MA</item>
        <item>NY</item>
    </one-of>
</rule>

<!-- "Boston MA" -> city = Boston, state = MA -->
<rule id="city_state" scope="public">
    <ruleref uri="#city"/> <ruleref uri="#state"/>
</rule>
```

Séquences et encapsulation

L'exemple suivant montre les séquences prises en charge. Pour plus d'informations, voir Séquences et encapsulation dans la recommandation W3C de la version 1 de la spécification grammaticale de reconnaissance vocale.

Example (Exemple)

```
<!-- sequence of tokens -->
```

```
this is a test

<!--sequence of rule references-->
<ruleref uri="#action"/> <ruleref uri="#object"/>

<!--sequence of tokens and rule references-->
the <ruleref uri="#object"/> is <ruleref uri="#color"/>

<!-- sequence container -->
<item>fly to <ruleref uri="#city"/> </item>
```

Répète

Le tableau suivant présente les extensions répétées prises en charge pour les règles. Pour plus d'informations, consultez la section Répétitions dans la version 1 de la recommandation du W3C de la spécification grammaticale de reconnaissance vocale.

formulaire XML Exemple	Attitude	Pris en charge ?
repeat="n » repeat="6 »	L'expression contenue est répétée exactement « n » fois. « n » doit être « 0 » ou un entier positif.	Oui
repeat="m-n » répéter « 4-6 »	L'expansion contenue est répétée entre « m » et « n » fois (inclus). « m » et « n » doivent tous deux être « 0 » ou un entier positif, et « m » doit être inférieur ou égal à « n ».	Oui
répète="m- » répéter « 3- »	L'expansion contenue est répétée « m » fois ou plus (inclus). « m » doit être « 0 » ou un entier positif. Par exemple, « 3- » indique que l'expansion contenue peut se	Oui

formulaire XML Exemple	Attitude	Pris en charge ?
	produire trois, quatre, cinq fois ou plus.	
répéter="0-1"	L'extension contenue est facultative.	Oui
<item repeat="2-4" repeat-prob="0.8">		Non

Langage

La discussion suivante s'applique aux identificateurs de langue appliqués aux grammaires. Pour plus d'informations, voir Langue dans la version 1 de la recommandation W3C de la spécification grammaticale de reconnaissance vocale.

Par défaut, une grammaire est un document en une seule langue dont l'identifiant de langue est fourni dans la déclaration de langue figurant dans l'en-tête de grammaire. Tous les jetons de cette grammaire, sauf indication contraire, seront gérés conformément à la langue de la grammaire. Les déclarations linguistiques au niveau de la grammaire ne sont pas prises en charge.

Dans l'exemple suivant :

1. La déclaration d'en-tête grammaticale pour la langue « en-US » est prise en charge par Amazon Lex V2.
2. L'attachement linguistique au niveau de l'article (*surligné en rouge*) n'est pas pris en charge. Amazon Lex V2 génère une erreur de validation si la langue d'une pièce jointe est différente de la déclaration d'en-tête.

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE grammar PUBLIC "-//W3C//DTD GRAMMAR 1.0//EN"
                  "http://www.w3.org/TR/speech-grammar/grammar.dtd">

<!-- the default grammar language is US English -->
<grammar xmlns="http://www.w3.org/2001/06/grammar"
```

```
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                            http://www.w3.org/TR/speech-grammar/grammar.xsd"
        xml:lang="en-US" version="1.0">

  <!--
     single language attachment to tokens
     "yes" inherits US English language
     "oui" is Canadian French language
  -->
  <rule id="yes">
    <one-of>
      <item>yes</item>
      <item xml:lang="fr-CA">oui</item>
    </one-of>
  </rule>

  <!-- Single language attachment to an expansion -->
  <rule id="people1">
    <one-of xml:lang="fr-CA">
      <item>Michel Tremblay</item>
      <item>André Roy</item>
    </one-of>
  </rule>
</grammar>
```

Étiquettes

La discussion suivante s'applique aux balises définies pour les grammaires. Pour plus d'informations, voir les balises dans la recommandation du W3C de la version 1 de la spécification grammaticale de reconnaissance vocale.

Sur la base de la spécification SRGS, les balises peuvent être définies de la manière suivante :

1. Dans le cadre d'une déclaration d'en-tête telle que décrite dansDéclarations d'en-tête.
2. Dans le cadre d'une <rule>définition.

Les formats de balises suivants sont pris en charge :

- `semantics/1.0`(SISR, ECMAScript)
- `semantics/1.0-literals`(Littéraux de chaîne SISR)

Les formats de balises suivants ne sont pas pris en charge :

- `swi-semantics/1.0`(Propriété de Nuance)

Example (Exemple)

```
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xml:base="http://www.example.com/base-file-path"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US"
         version="1.0"
         mode="voice"
         root="city"
         tag-format="semantics/1.0-literals">
    <rule id="no">
        <one-of>
            <item>no</item>
            <item>nope</item>
            <item>no way</item>
        </one-of>
        <tag>no</tag>
    </rule>
</grammar>
```

Poids

Vous pouvez ajouter l'attribut de poids à un élément. Le poids est une valeur à virgule flottante positive qui représente le degré d'accentuation de la phrase de l'élément lors de la reconnaissance vocale. Pour plus d'informations, voir les pondérations dans la version 1 de la recommandation du W3C de la spécification grammaticale de reconnaissance vocale.

Les pondérations doivent être supérieures à 0 et inférieures ou égales à 10, et ne peuvent comporter qu'une décimale. Si la pondération est supérieure à 0 et inférieure à 1, la phrase est accentuée négativement. Si le poids est supérieur à 1 et inférieur ou égal à 10, la phrase est renforcée positivement. Une pondération de 1 équivaut à ne pas donner de poids du tout, et il n'y a pas de renforcement pour la phrase.

Attribuer des pondérations appropriées aux éléments pour améliorer les performances de reconnaissance vocale est une tâche difficile. Voici quelques conseils que vous pouvez suivre pour attribuer des poids :

- Commencez par une grammaire sans qu'aucune pondération ne soit attribuée aux éléments.
- Déterminez quels modèles du discours sont fréquemment mal identifiés.
- Appliquez des valeurs de pondération différentes jusqu'à ce que vous remarquiez une amélioration des performances de reconnaissance vocale et qu'il n'y ait aucune régression.

Exemple 1

Par exemple, si vous avez une grammaire pour les aéroports et que vous constatez que New York est souvent identifiée à tort comme Newark, vous pouvez améliorer New York de manière positive en lui attribuant une pondération de 5.

```
<rule> id="airport">
    <one-of>
        <item>
            Boston
            <tag>out="Boston"</tag>
        </item>
        <item weight="5">
            New York
            <tag>out="New York"</tag>
        </item>
        <item>
            Newark
            <tag>out="Newark"</tag>
        </item>
    </one-of>
</rule>
```

Exemple 2

Par exemple, vous avez une grammaire pour le code de réservation de la compagnie aérienne qui commence par un alphabet anglais suivi de trois chiffres. Le code de réservation commence très probablement par B ou D, mais vous remarquez que B est souvent identifié à tort comme P et D comme T. Vous pouvez augmenter positivement B et D.

```
<rule> id="alphabet">
    <one-of>
        <item>A<tag>out.letters+='A';</tag></item>
        <item weight="3.5">B<tag>out.letters+='B';</tag></item>
        <item>C<tag>out.letters+='C';</tag></item>
        <item weight="2.9">D<tag>out.letters+='D';</tag></item>
        <item>E<tag>out.letters+='E';</tag></item>
        <item>F<tag>out.letters+='F';</tag></item>
        <item>G<tag>out.letters+='G';</tag></item>
        <item>H<tag>out.letters+='H';</tag></item>
        <item>I<tag>out.letters+='I';</tag></item>
        <item>J<tag>out.letters+='J';</tag></item>
        <item>K<tag>out.letters+='K';</tag></item>
        <item>L<tag>out.letters+='L';</tag></item>
        <item>M<tag>out.letters+='M';</tag></item>
        <item>N<tag>out.letters+='N';</tag></item>
        <item>O<tag>out.letters+='O';</tag></item>
        <item>P<tag>out.letters+='P';</tag></item>
        <item>Q<tag>out.letters+='Q';</tag></item>
        <item>R<tag>out.letters+='R';</tag></item>
        <item>S<tag>out.letters+='S';</tag></item>
        <item>T<tag>out.letters+='T';</tag></item>
        <item>U<tag>out.letters+='U';</tag></item>
        <item>V<tag>out.letters+='V';</tag></item>
        <item>W<tag>out.letters+='W';</tag></item>
        <item>X<tag>out.letters+='X';</tag></item>
        <item>Y<tag>out.letters+='Y';</tag></item>
        <item>Z<tag>out.letters+='Z';</tag></item>
    </one-of>
</rule>
```

Format de script

Amazon Lex V2 prend en charge les fonctionnalités ECMAScript suivantes pour définir des grammaires.

Amazon Lex V2 prend en charge les fonctionnalités ECMAScript suivantes lors de la spécification de balises dans la grammaire. `tag-format`doit être envoyé à `semantics/1.0` lorsque des balises ECMAScript sont utilisées dans la grammaire. Pour plus d'informations, consultez la spécification du langage ECMA-262 ECMAScript 2021.

```
<grammar version="1.0"
xmlns="http://www.w3.org/2001/06/grammar"
xml:lang="en-US"
tag-format="semantics/1.0"
root="card_number">
```

Rubriques

- Déclaration variable
- Expressions
- Si déclaration
- Déclaration Switch
- Déclarations de fonctions
- Déclaration d'itération
- Déclaration de blocage
- Commentaires
- Déclarations non étayées

Ce document contient du matériel issu de la norme ECMAScript (disponible à l'adresse https://www.ecma-international.org/ publications-and-standards /standards/ecma-262/). Le document de spécification du langage ECMAScript est disponible auprès d'Ecma International sous la licence suivante.

Texte de licence

(iii) des traductions de ce document dans des langues autres que l'anglais et dans différents formats et

(iv) fonctionne en utilisant cette spécification dans des produits conformes à la norme en mettant en œuvre (par exemple par copier-coller en tout ou en partie) les fonctionnalités qu'elle contient.

Toutefois, le contenu de ce document lui-même ne peut être modifié de quelque manière que ce soit, y compris en supprimant la notice de copyright ou les références à Ecma International, sauf si cela est nécessaire pour le traduire dans des langues autres que l'anglais ou dans un format différent.

La version officielle d'un document d'Ecma International est la version en langue anglaise disponible sur le site Web d'Ecma International. En cas de divergence entre une version traduite et la version officielle, la version officielle prévaut.

Les autorisations limitées accordées ci-dessus sont perpétuelles et ne seront pas révoquées par Ecma International ou ses successeurs ou ayants droit. Ce document et les informations qu'il contient sont fournis « TELS QUELS » et ECMA INTERNATIONAL DÉCLINE TOUTE GARANTIE, EXPRESSE OU IMPLICITE, Y COMPRIS, MAIS SANS S'Y LIMITER, TOUTE GARANTIE SELON LAQUELLE L'UTILISATION DES INFORMATIONS QU'IL CONTIENT N'ENFREINDRA AUCUN DROIT DE PROPRIÉTÉ NI AUCUNE GARANTIE IMPLICITE DE QUALITÉ MARCHANDE OU D'ADÉQUATION À UN USAGE PARTICULIER. »

Déclaration variable

Une instruction variable définit une ou plusieurs variables.

```
var x = 10;
var x = 10, var y = <expression>;
```

Expressions

Type d'expression	Syntaxe	Exemple	Pris en charge ?
Expression régulière littérale	Chaîne littérale contenant des caractères spéciaux regex valides	`"^\d\.$"`	Non
Fonction	`function functionN`	`var x = function calc() {`	Non

Type d'expression	Syntaxe	Exemple	Pris en charge ?
	`ame(param eters) { functionBody}`	`return 10; }`	
Delete	`delete expressio n`	`delete obj.prope rty;`	Non
Void	`void expression`	`void (2 == '2');`	Non
Type de	`typeof expressio n`	`typeof 42;`	Non
Index des membres	`expression [ expressions ]`	`var fruits = ["apple"]; fruits[0];`	Oui
Point de membre	`expression . identifier`	`out.value`	oui
Arguments	`expression (arguments)`	`new Date('199 4-10-11')`	Oui
Après l'incrémentation	`expression++`	`var x=10; x++;`	Oui
Après le décret	`expression--`	`var x=10; x--;`	Oui
Pré-incrémentation	`++expression`	`var x=10; ++x;`	Oui
Avant le décrément	`--expression`	`var x=10; --x;`	Oui
Unaire plus/Unaire moins	`+expression / - expression`	`+x / -x;`	Oui

Type d'expression	Syntaxe	Exemple	Pris en charge ?
Mais pas	`~ expression`	`const a = 5; consol e.log( ~a );`	Oui
Logique, non.	`! expression`	`!(a > 0 || b > 0)`	Oui
Multiplicatif	`expression ('*' | '/' | '%') expression`	`(x + y) * (a / b)`	Oui
Additif	`expression ('+' | '-') expressio n`	`(a + b) - (a - (a + b))`	Oui
Déplacement de bits	`expression ('<<' | '>>' | '>>>') expressio n`	`(a >> b) >>> c`	Oui
Relatif	`expression ('<' | '>' | '<=' | '>=') expressio n`	`if (a > b) { ... }`	Oui
Entrée	`expression in expression`	`fruits[0] in otherFruits;`	Oui
Égalité	`expression ('==' | '!=' | '===' | '!==') expression`	`if (a == b) { ... }`	Oui
Bit et/xor/ou	`expression ('&' | '^' | '|') expression`	`a & b / a ^ b / a | b`	Oui

Type d'expression	Syntaxe	Exemple	Pris en charge ?
Logique et/ ou	`expression ('&&' | '||') expression`	`if (a && (b ||c)) { ...}`	Oui
Ternaire	`expression ? expression : expression`	`a > b ? obj.prop : 0`	Oui
Affectation	`expression = expression`	`out.value = "string";`	Oui
Opérateur d'assignation	`expression ('*=' | '/=' | '+=' | '-=' | '%=') expression`	`a *= 10;`	Oui
Opérateur d'affectation bit par bit	`expression ('<<=' | '>>=' | '>>>=' | '&=' | '^=' | '|=') expression`	`a <<= 10;`	Oui
Identifiant	`identifierSequence` où IdentifierSequence est une séquence de caractères valides	`fruits=[10, 20, 30];`	Oui
Littéral nul	`null`	`x = null;`	Oui
Littéral booléen	`true | false`	`x = true;`	Oui

Type d'expression	Syntaxe	Exemple	Pris en charge ?
Chaîne littérale	`'string' / "string"`	`a = 'hello', b = "world";`	Oui
Littéral décimal	`integer [.] digits [exponent ]`	`111.11 e+12`	Oui
Littéral hexadécimal	`0 (x | X)[0-9a-f A-F]`	`0x123ABC`	Oui
Littéral octal	`0 [0-7]`	`"051"`	Oui
Tableau littéral	`[ expressio n, ... ]`	`v = [a, b, c];`	Oui
Objet littéral	`{property: value, ...}`	`out = {value: 1, flag: false};`	Oui
Entre parenthèses	`( expressions )`	`x + (x + y)`	Oui

Si déclaration

```
if (expressions) {
    statements;
} else {
    statements;
}
```

Remarque : Dans l'exemple précédent, `expressions` `statements` il doit s'agir de l'un des modèles pris en charge par ce document.

Déclaration Switch

```
switch (expression) {
    case (expression):
```

```
     statements
     .
     .
     .
   default:
      statements
}
```

Remarque : Dans l'exemple précédent, `expressions` `statements` il doit s'agir de l'un des modèles pris en charge par ce document.

Déclarations de fonctions

```
function functionIdentifier([parameterList, ...]) {
     <function body>
}
```

Déclaration d'itération

Les instructions d'itération peuvent prendre l'une des formes suivantes :

```
// Do..While statement
do {
   statements
} while (expressions)


// While Loop
while (expressions) {
   statements
}

// For Loop
for ([initialization]; [condition]; [final-expression])
   statement

// For..In
for (variable in object) {
  statement
}
```

Déclaration de blocage

```
{
   statements
}

// Example
{
    x = 10;
   if (x > 10) {
     console.log("greater than 10");
   }
}
```

Remarque : Dans l'exemple précédent, `statements` le bloc doit figurer parmi ceux pris en charge par ce document.

Commentaires

```
// Single Line Comments
"// <comment>"

// Multineline comments
/**
<comment>
**/
```

Déclarations non étayées

Amazon Lex V2 ne prend pas en charge les fonctionnalités ECMAScript suivantes.

Rubriques

- Déclaration vide
- Continuer la déclaration
- Déclaration de rupture
- Déclaration de retour
- Déclaration de lancer
- Essayez la déclaration

- Déclaration du Debugger
- Déclaration étiquetée
- Déclaration de classe

Déclaration vide

L'instruction vide est utilisée pour ne fournir aucune instruction. La syntaxe d'une instruction vide est la suivante :

```
;
```

Continuer la déclaration

L'instruction continue sans étiquette est compatible avec leDéclaration d'itération. L'instruction continue avec une étiquette n'est pas prise en charge.

```
// continue with label
// this allows the program to jump to a
// labelled statement (see labelled statement below)
continue <label>;
```

Déclaration de rupture

L'instruction break sans étiquette est compatible avec leDéclaration d'itération. L'instruction break associée à une étiquette n'est pas prise en charge.

```
// break with label
// this allows the program to break out of a
// labelled statement (see labelled statement below)
break <label>;
```

Déclaration de retour

```
return expression;
```

Déclaration de lancer

L'instruction throw est utilisée pour lancer une exception définie par l'utilisateur.

```
throw expression;
```

Essayez la déclaration

```
try {
  statements
}
catch (expression) {
  statements
}
finally {
  statements
}
```

Déclaration du Debugger

L'instruction debugger est utilisée pour invoquer la fonctionnalité de débogage fournie par l'environnement.

```
debugger;
```

Déclaration étiquetée

L'instruction étiquetée peut être utilisée avec des `continue` instructions `break` or.

```
label:
   statements


// Example
let str = '';

loop1:
for (let i = 0; i < 5; i++) {
  if (i === 1) {
    continue loop1;
  }
  str = str + i;
}

console.log(str);
```

Déclaration de classe

```
class Rectangle {
  constructor(height, width) {
    this.height = height;
    this.width = width;
  }
}
```

Grammaires sectorielles

Les grammaires industrielles sont un ensemble de fichiers XML à utiliser avec le type d'emplacement grammatical. Vous pouvez les utiliser pour offrir rapidement une expérience cohérente à l'utilisateur final lorsque vous migrez des flux de travail de réponse vocale interactifs vers Amazon Lex V2. Vous pouvez choisir parmi une gamme de grammaires prédéfinies dans trois domaines : services financiers, assurances et télécommunications. Il existe également un ensemble générique de grammaires que vous pouvez utiliser comme point de départ pour vos propres grammaires.

Les grammaires contiennent les règles de collecte des informations et les balises ECMAScript pour l'interprétation sémantique.

Grammaires pour les services financiers (téléchargement)

Les grammaires suivantes sont prises en charge pour les services financiers : numéros de compte et de routage, numéros de carte de crédit et de prêt, dossier de solvabilité, dates d'ouverture et de fermeture du compte et numéro de sécurité sociale.

Numéro de compte

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">


        <!-- Test Cases
```

```
        Grammar will support the following inputs:

            Scenario 1:
                Input: My account number is A B C 1 2 3 4
                Output: ABC1234

            Scenario 2:
                Input: My account number is 1 2 3 4 A B C
                Output: 1234ABC

            Scenario 3:
                Input: Hmm My account number is 1 2 3 4 A B C 1
                Output: 123ABC1
        -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item><ruleref uri="#alphanumeric"/><tag>out +=
rules.alphanumeric.alphanum;</tag></item>
            <item repeat="0-1"><ruleref uri="#alphabets"/><tag>out +=
rules.alphabets.letters;</tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
rules.digits.numbers</tag></item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
            <one-of>
                <item repeat="0-1">account number is</item>
                <item repeat="0-1">Account Number</item>
                <item repeat="0-1">Here is my Account Number </item>
                <item repeat="0-1">Yes, It is</item>
                <item repeat="0-1">Yes It is</item>
                <item repeat="0-1">Yes It's</item>
                <item repeat="0-1">My account Id is</item>
                <item repeat="0-1">This is the account Id</item>
                <item repeat="0-1">account Id</item>
            </one-of>
        </rule>

        <rule id="hesitation">
          <one-of>
             <item>Hmm</item>
             <item>Mmm</item>
```

```
            <item>My</item>
          </one-of>
        </rule>

        <rule id="alphanumeric" scope="public">
            <tag>out.alphanum=""</tag>
            <item><ruleref uri="#alphabets"/><tag>out.alphanum +=
rules.alphabets.letters;</tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.alphanum +=
rules.digits.numbers</tag></item>
        </rule>

        <rule id="alphabets">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.letters=""</tag>
            <tag>out.firstOccurence=""</tag>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.firstOccurence +=
rules.digits.numbers; out.letters += out.firstOccurence;</tag></item>
            <item repeat="1-">
                <one-of>
                    <item>A<tag>out.letters+='A';</tag></item>
                    <item>B<tag>out.letters+='B';</tag></item>
                    <item>C<tag>out.letters+='C';</tag></item>
                    <item>D<tag>out.letters+='D';</tag></item>
                    <item>E<tag>out.letters+='E';</tag></item>
                    <item>F<tag>out.letters+='F';</tag></item>
                    <item>G<tag>out.letters+='G';</tag></item>
                    <item>H<tag>out.letters+='H';</tag></item>
                    <item>I<tag>out.letters+='I';</tag></item>
                    <item>J<tag>out.letters+='J';</tag></item>
                    <item>K<tag>out.letters+='K';</tag></item>
                    <item>L<tag>out.letters+='L';</tag></item>
                    <item>M<tag>out.letters+='M';</tag></item>
                    <item>N<tag>out.letters+='N';</tag></item>
                    <item>O<tag>out.letters+='O';</tag></item>
                    <item>P<tag>out.letters+='P';</tag></item>
                    <item>Q<tag>out.letters+='Q';</tag></item>
                    <item>R<tag>out.letters+='R';</tag></item>
                    <item>S<tag>out.letters+='S';</tag></item>
                    <item>T<tag>out.letters+='T';</tag></item>
                    <item>U<tag>out.letters+='U';</tag></item>
                    <item>V<tag>out.letters+='V';</tag></item>
                    <item>W<tag>out.letters+='W';</tag></item>
                    <item>X<tag>out.letters+='X';</tag></item>
```

```
                    <item>Y<tag>out.letters+='Y';</tag></item>
                    <item>Z<tag>out.letters+='Z';</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="digits">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.numbers=""</tag>
            <item repeat="1-10">
                <one-of>
                    <item>0<tag>out.numbers+=0;</tag></item>
                    <item>1<tag>out.numbers+=1;</tag></item>
                    <item>2<tag>out.numbers+=2;</tag></item>
                    <item>3<tag>out.numbers+=3;</tag></item>
                    <item>4<tag>out.numbers+=4;</tag></item>
                    <item>5<tag>out.numbers+=5;</tag></item>
                    <item>6<tag>out.numbers+=6;</tag></item>
                    <item>7<tag>out.numbers+=7;</tag></item>
                    <item>8<tag>out.numbers+=8;</tag></item>
                    <item>9<tag>out.numbers+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

Code d'acheminement

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="digits"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
```

```
                Input: My routing number is 1 2 3 4 5 6 7 8 9
                Output: 123456789

             Scenario 2:
                Input: routing number 1 2 3 4 5 6 7 8 9
                Output: 123456789

         -->

        <rule id="digits">
            <tag>out=""</tag>
            <item><ruleref uri="#singleDigit"/><tag>out += rules.singleDigit.digit;</
tag></item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">My routing number</item>
              <item repeat="0-1">Routing number of</item>
              <item repeat="0-1">The routing number is</item>
           </one-of>
        </rule>

        <rule id="hesitation">
            <one-of>
               <item>Hmm</item>
               <item>Mmm</item>
               <item>My</item>
            </one-of>
          </rule>

        <rule id="singleDigit">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.digit=""</tag>
            <item repeat="16">
                <one-of>
                    <item>0<tag>out.digit+=0;</tag></item>
                    <item>zero<tag>out.digit+=0;</tag></item>
                    <item>1<tag>out.digit+=1;</tag></item>
                    <item>one<tag>out.digit+=1;</tag></item>
                    <item>2<tag>out.digit+=2;</tag></item>
                    <item>two<tag>out.digit+=2;</tag></item>
                    <item>3<tag>out.digit+=3;</tag></item>
```

```
                    <item>three<tag>out.digit+=3;</tag></item>
                    <item>4<tag>out.digit+=4;</tag></item>
                    <item>four<tag>out.digit+=4;</tag></item>
                    <item>5<tag>out.digit+=5;</tag></item>
                    <item>five<tag>out.digit+=5;</tag></item>
                    <item>6<tag>out.digit+=6;</tag></item>
                    <item>six<tag>out.digit+=5;</tag></item>
                    <item>7<tag>out.digit+=7;</tag></item>
                    <item>seven<tag>out.digit+=7;</tag></item>
                    <item>8<tag>out.digit+=8;</tag></item>
                    <item>eight<tag>out.digit+=8;</tag></item>
                    <item>9<tag>out.digit+=9;</tag></item>
                    <item>nine<tag>out.digit+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

Numéro de carte de crédit

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="digits"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: My credit card number is 1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7
                 Output: 1234567891234567

             Scenario 2:
                 Input: card number 1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7
                 Output: 1234567891234567

         -->
```

```
        <rule id="digits">
            <tag>out=""</tag>
            <item><ruleref uri="#singleDigit"/><tag>out += rules.singleDigit.digit;</
tag></item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">My credit card number is</item>
              <item repeat="0-1">card number</item>
           </one-of>
        </rule>

        <rule id="hesitation">
            <one-of>
               <item>Hmm</item>
               <item>Mmm</item>
               <item>My</item>
            </one-of>
          </rule>

        <rule id="singleDigit">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.digit=""</tag>
            <item repeat="16">
                <one-of>
                    <item>0<tag>out.digit+=0;</tag></item>
                    <item>zero<tag>out.digit+=0;</tag></item>
                    <item>1<tag>out.digit+=1;</tag></item>
                    <item>one<tag>out.digit+=1;</tag></item>
                    <item>2<tag>out.digit+=2;</tag></item>
                    <item>two<tag>out.digit+=2;</tag></item>
                    <item>3<tag>out.digit+=3;</tag></item>
                    <item>three<tag>out.digit+=3;</tag></item>
                    <item>4<tag>out.digit+=4;</tag></item>
                    <item>four<tag>out.digit+=4;</tag></item>
                    <item>5<tag>out.digit+=5;</tag></item>
                    <item>five<tag>out.digit+=5;</tag></item>
                    <item>6<tag>out.digit+=6;</tag></item>
                    <item>six<tag>out.digit+=5;</tag></item>
                    <item>7<tag>out.digit+=7;</tag></item>
                    <item>seven<tag>out.digit+=7;</tag></item>
```

```
                    <item>8<tag>out.digit+=8;</tag></item>
                    <item>eight<tag>out.digit+=8;</tag></item>
                    <item>9<tag>out.digit+=9;</tag></item>
                    <item>nine<tag>out.digit+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

ID de prêt

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">


        <!-- Test Cases

        Grammar will support the following inputs:

            Scenario 1:
                Input: My loan Id is A B C 1 2 3 4
                Output: ABC1234
        -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item><ruleref uri="#alphanumeric"/><tag>out +=
 rules.alphanumeric.alphanum;</tag></item>
            <item repeat="0-1"><ruleref uri="#alphabets"/><tag>out +=
 rules.alphabets.letters;</tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
 rules.digits.numbers</tag></item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
```

```
            <one-of>
                <item repeat="0-1">my loan number is</item>
                <item repeat="0-1">The loan number</item>
                <item repeat="0-1">The loan is </item>
                <item repeat="0-1">The number is</item>
                <item repeat="0-1">loan number</item>
                <item repeat="0-1">loan number of</item>
                <item repeat="0-1">loan Id is</item>
                <item repeat="0-1">My loan Id is</item>
            </one-of>
        </rule>

        <rule id="hesitation">
          <one-of>
             <item>Hmm</item>
             <item>Mmm</item>
             <item>My</item>
          </one-of>
        </rule>

        <rule id="alphanumeric" scope="public">
            <tag>out.alphanum=""</tag>
            <item><ruleref uri="#alphabets"/><tag>out.alphanum +=
rules.alphabets.letters;</tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.alphanum +=
rules.digits.numbers</tag></item>
        </rule>

        <rule id="alphabets">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.letters=""</tag>
            <tag>out.firstOccurence=""</tag>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.firstOccurence +=
rules.digits.numbers; out.letters += out.firstOccurence;</tag></item>
            <item repeat="1-">
                <one-of>
                    <item>A<tag>out.letters+='A';</tag></item>
                    <item>B<tag>out.letters+='B';</tag></item>
                    <item>C<tag>out.letters+='C';</tag></item>
                    <item>D<tag>out.letters+='D';</tag></item>
                    <item>E<tag>out.letters+='E';</tag></item>
                    <item>F<tag>out.letters+='F';</tag></item>
                    <item>G<tag>out.letters+='G';</tag></item>
                    <item>H<tag>out.letters+='H';</tag></item>
```

```
                    <item>I<tag>out.letters+='I';</tag></item>
                    <item>J<tag>out.letters+='J';</tag></item>
                    <item>K<tag>out.letters+='K';</tag></item>
                    <item>L<tag>out.letters+='L';</tag></item>
                    <item>M<tag>out.letters+='M';</tag></item>
                    <item>N<tag>out.letters+='N';</tag></item>
                    <item>O<tag>out.letters+='O';</tag></item>
                    <item>P<tag>out.letters+='P';</tag></item>
                    <item>Q<tag>out.letters+='Q';</tag></item>
                    <item>R<tag>out.letters+='R';</tag></item>
                    <item>S<tag>out.letters+='S';</tag></item>
                    <item>T<tag>out.letters+='T';</tag></item>
                    <item>U<tag>out.letters+='U';</tag></item>
                    <item>V<tag>out.letters+='V';</tag></item>
                    <item>W<tag>out.letters+='W';</tag></item>
                    <item>X<tag>out.letters+='X';</tag></item>
                    <item>Y<tag>out.letters+='Y';</tag></item>
                    <item>Z<tag>out.letters+='Z';</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="digits">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.numbers=""</tag>
            <item repeat="1-10">
                <one-of>
                    <item>0<tag>out.numbers+=0;</tag></item>
                    <item>1<tag>out.numbers+=1;</tag></item>
                    <item>2<tag>out.numbers+=2;</tag></item>
                    <item>3<tag>out.numbers+=3;</tag></item>
                    <item>4<tag>out.numbers+=4;</tag></item>
                    <item>5<tag>out.numbers+=5;</tag></item>
                    <item>6<tag>out.numbers+=6;</tag></item>
                    <item>7<tag>out.numbers+=7;</tag></item>
                    <item>8<tag>out.numbers+=8;</tag></item>
                    <item>9<tag>out.numbers+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

Cote de crédit

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: The number is fifteen
                 Output: 15

             Scenario 2:
                 Input: My credit score is fifteen
                 Output: 15
         -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <one-of>
              <item repeat="1"><ruleref uri="#digits"/><tag>out+= rules.digits;</tag></
item>
              <item repeat="1"><ruleref uri="#teens"/><tag>out+= rules.teens;</tag></
item>
              <item repeat="1"><ruleref uri="#above_twenty"/><tag>out+=
 rules.above_twenty;</tag></item>
            </one-of>
        </rule>

        <rule id="text">
           <one-of>
              <item repeat="0-1">Credit score is</item>
              <item repeat="0-1">Last digits are</item>
              <item repeat="0-1">The number is</item>
              <item repeat="0-1">That's</item>
              <item repeat="0-1">It is</item>
```

```
            <item repeat="0-1">My credit score is</item>
        </one-of>
    </rule>

    <rule id="digits">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>0<tag>out=0;</tag></item>
            <item>1<tag>out=1;</tag></item>
            <item>2<tag>out=2;</tag></item>
            <item>3<tag>out=3;</tag></item>
            <item>4<tag>out=4;</tag></item>
            <item>5<tag>out=5;</tag></item>
            <item>6<tag>out=6;</tag></item>
            <item>7<tag>out=7;</tag></item>
            <item>8<tag>out=8;</tag></item>
            <item>9<tag>out=9;</tag></item>
            <item>one<tag>out=1;</tag></item>
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>

    <rule id="teens">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>ten<tag>out=10;</tag></item>
            <item>eleven<tag>out=11;</tag></item>
            <item>twelve<tag>out=12;</tag></item>
            <item>thirteen<tag>out=13;</tag></item>
            <item>fourteen<tag>out=14;</tag></item>
            <item>fifteen<tag>out=15;</tag></item>
            <item>sixteen<tag>out=16;</tag></item>
            <item>seventeen<tag>out=17;</tag></item>
            <item>eighteen<tag>out=18;</tag></item>
            <item>nineteen<tag>out=19;</tag></item>
            <item>10<tag>out=10;</tag></item>
            <item>11<tag>out=11;</tag></item>
```

```
                <item>12<tag>out=12;</tag></item>
                <item>13<tag>out=13;</tag></item>
                <item>14<tag>out=14;</tag></item>
                <item>15<tag>out=15;</tag></item>
                <item>16<tag>out=16;</tag></item>
                <item>17<tag>out=17;</tag></item>
                <item>18<tag>out=18;</tag></item>
                <item>19<tag>out=19;</tag></item>
            </one-of>
        </rule>

        <rule id="above_twenty">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
                <item>forty<tag>out=40;</tag></item>
                <item>fifty<tag>out=50;</tag></item>
                <item>sixty<tag>out=60;</tag></item>
                <item>seventy<tag>out=70;</tag></item>
                <item>eighty<tag>out=80;</tag></item>
                <item>ninety<tag>out=90;</tag></item>
                <item>20<tag>out=20;</tag></item>
                <item>30<tag>out=30;</tag></item>
                <item>40<tag>out=40;</tag></item>
                <item>50<tag>out=50;</tag></item>
                <item>60<tag>out=60;</tag></item>
                <item>70<tag>out=70;</tag></item>
                <item>80<tag>out=80;</tag></item>
                <item>90<tag>out=90;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>

</grammar>
```

Date d'ouverture du compte

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
```

```
                              http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: I opened account on July Two Thousand and Eleven
                 Output: 07/11

             Scenario 2:
                 Input: I need account number opened on July Two Thousand and Eleven
                 Output: 07/11

         -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item repeat="1-10">
                <item repeat="1"><ruleref uri="#months"/><tag>out = out +
 rules.months.mon + "/";</tag></item>
                <one-of>
                    <item><ruleref uri="#thousands"/><tag>out += rules.thousands;</
tag></item>
                    <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
 rules.digits;</tag></item>
                    <item repeat="0-1"><ruleref uri="#teens"/><tag>out +=
 rules.teens;</tag></item>
                    <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty;</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">I opened account on </item>
              <item repeat="0-1">I need account number opened on </item>
           </one-of>
```

```
        </rule>

        <rule id="hesitation">
            <one-of>
               <item>Hmm</item>
               <item>Mmm</item>
               <item>My</item>
            </one-of>
          </rule>
        <rule id="months">
           <item repeat="0-1"><ruleref uri="#text"/></item>
           <tag>out.mon=""</tag>
           <one-of>
               <item>january<tag>out.mon+="01";</tag></item>
               <item>february<tag>out.mon+="02";</tag></item>
               <item>march<tag>out.mon+="03";</tag></item>
               <item>april<tag>out.mon+="04";</tag></item>
               <item>may<tag>out.mon+="05";</tag></item>
               <item>june<tag>out.mon+="06";</tag></item>
               <item>july<tag>out.mon+="07";</tag></item>
               <item>august<tag>out.mon+="08";</tag></item>
               <item>september<tag>out.mon+="09";</tag></item>
               <item>october<tag>out.mon+="10";</tag></item>
               <item>november<tag>out.mon+="11";</tag></item>
               <item>december<tag>out.mon+="12";</tag></item>
               <item>jan<tag>out.mon+="01";</tag></item>
               <item>feb<tag>out.mon+="02";</tag></item>
               <item>aug<tag>out.mon+="08";</tag></item>
               <item>sept<tag>out.mon+="09";</tag></item>
               <item>oct<tag>out.mon+="10";</tag></item>
               <item>nov<tag>out.mon+="11";</tag></item>
               <item>dec<tag>out.mon+="12";</tag></item>
           </one-of>
        </rule>

        <rule id="digits">
            <one-of>
                <item>zero<tag>out=0;</tag></item>
                <item>one<tag>out=1;</tag></item>
                <item>two<tag>out=2;</tag></item>
                <item>three<tag>out=3;</tag></item>
                <item>four<tag>out=4;</tag></item>
                <item>five<tag>out=5;</tag></item>
                <item>six<tag>out=6;</tag></item>
```

```
                <item>seven<tag>out=7;</tag></item>
                <item>eight<tag>out=8;</tag></item>
                <item>nine<tag>out=9;</tag></item>
            </one-of>
        </rule>

        <rule id="teens">
            <one-of>
                <item>ten<tag>out=10;</tag></item>
                <item>eleven<tag>out=11;</tag></item>
                <item>twelve<tag>out=12;</tag></item>
                <item>thirteen<tag>out=13;</tag></item>
                <item>fourteen<tag>out=14;</tag></item>
                <item>fifteen<tag>out=15;</tag></item>
                <item>sixteen<tag>out=16;</tag></item>
                <item>seventeen<tag>out=17;</tag></item>
                <item>eighteen<tag>out=18;</tag></item>
                <item>nineteen<tag>out=19;</tag></item>
            </one-of>
        </rule>

        <rule id="thousands">
            <item>two thousand<!--<tag>out=2000;</tag>--></item>
            <item repeat="0-1">and</item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out = rules.digits;</tag></
item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out = rules.teens;</tag></
item>
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out =
 rules.above_twenty;</tag></item>
        </rule>

        <rule id="above_twenty">
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
                <item>forty<tag>out=40;</tag></item>
                <item>fifty<tag>out=50;</tag></item>
                <item>sixty<tag>out=60;</tag></item>
                <item>seventy<tag>out=70;</tag></item>
                <item>eighty<tag>out=80;</tag></item>
                <item>ninety<tag>out=90;</tag></item>
            </one-of>
```

```
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
</grammar>
```

Date de paie automatique

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: I want to schedule auto pay for twenty five Dollar
                 Output: $25

             Scenario 2:
                 Input: Setup automatic payments for twenty five dollars
                 Output: $25

         -->

        <rule id="main" scope="public">
            <tag>out="$"</tag>
            <one-of>
                <item><ruleref uri="#sub_hundred"/><tag>out += rules.sub_hundred.sh;</
tag></item>
                <item><ruleref uri="#subThousands"/><tag>out += rules.subThousands;</
tag></item>
            </one-of>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
```

```
            <one-of>
               <item repeat="0-1">I want to schedule auto pay for</item>
               <item repeat="0-1">Setup automatic payments for twenty five dollars</
item>
               <item repeat="0-1">Auto pay amount of</item>
               <item repeat="0-1">Set it up for</item>
            </one-of>
         </rule>

         <rule id="hesitation">
             <one-of>
                <item>Hmm</item>
                <item>Mmm</item>
                <item>My</item>
             </one-of>
           </rule>

         <rule id="digits">
             <item repeat="0-1"><ruleref uri="#text"/></item>
             <tag>out.num = 0;</tag>
             <one-of>
                 <item>0<tag>out.num+=0;</tag></item>
                 <item>1<tag>out.num+=1;</tag></item>
                 <item>2<tag>out.num+=2;</tag></item>
                 <item>3<tag>out.num+=3;</tag></item>
                 <item>4<tag>out.num+=4;</tag></item>
                 <item>5<tag>out.num+=5;</tag></item>
                 <item>6<tag>out.num+=6;</tag></item>
                 <item>7<tag>out.num+=7;</tag></item>
                 <item>8<tag>out.num+=8;</tag></item>
                 <item>9<tag>out.num+=9;</tag></item>
                 <item>one<tag>out.num+=1;</tag></item>
                 <item>two<tag>out.num+=2;</tag></item>
                 <item>three<tag>out.num+=3;</tag></item>
                 <item>four<tag>out.num+=4;</tag></item>
                 <item>five<tag>out.num+=5;</tag></item>
                 <item>six<tag>out.num+=6;</tag></item>
                 <item>seven<tag>out.num+=7;</tag></item>
                 <item>eight<tag>out.num+=8;</tag></item>
                 <item>nine<tag>out.num+=9;</tag></item>
             </one-of>
             <item repeat="0-1"><ruleref uri="#currency"/></item>
         </rule>
```

```
        <rule id="teens">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.teen = 0;</tag>
            <one-of>
                <item>ten<tag>out.teen+=10;</tag></item>
                <item>eleven<tag>out.teen+=11;</tag></item>
                <item>twelve<tag>out.teen+=12;</tag></item>
                <item>thirteen<tag>out.teen+=13;</tag></item>
                <item>fourteen<tag>out.teen+=14;</tag></item>
                <item>fifteen<tag>out.teen+=15;</tag></item>
                <item>sixteen<tag>out.teen+=16;</tag></item>
                <item>seventeen<tag>out.teen+=17;</tag></item>
                <item>eighteen<tag>out.teen+=18;</tag></item>
                <item>nineteen<tag>out.teen+=19;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>

        <rule id="above_twenty">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.tens = 0;</tag>
            <one-of>
                <item>twenty<tag>out.tens+=20;</tag></item>
                <item>thirty<tag>out.tens+=30;</tag></item>
                <item>forty<tag>out.tens+=40;</tag></item>
                <item>fifty<tag>out.tens+=50;</tag></item>
                <item>sixty<tag>out.tens+=60;</tag></item>
                <item>seventy<tag>out.tens+=70;</tag></item>
                <item>eighty<tag>out.tens+=80;</tag></item>
                <item>ninety<tag>out.tens+=90;</tag></item>
                <item>hundred<tag>out.tens+=100;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.tens +=
rules.digits.num;</tag></item>
        </rule>

        <rule id="currency">
            <one-of>
                <item repeat="0-1">dollars</item>
                <item repeat="0-1">Dollars</item>
                <item repeat="0-1">dollar</item>
                <item repeat="0-1">Dollar</item>
            </one-of>
```

```
        </rule>


        <rule id="sub_hundred">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.sh = 0;</tag>
            <one-of>
                <item><ruleref uri="#teens"/><tag>out.sh += rules.teens.teen;</tag></
item>
                <item>
                    <ruleref uri="#above_twenty"/><tag>out.sh +=
 rules.above_twenty.tens;</tag>
                </item>
                <item><ruleref uri="#digits"/><tag>out.sh += rules.digits.num;</tag></
item>
            </one-of>
         </rule>

        <rule id="subThousands">
            <ruleref uri="#sub_hundred"/><tag>out = (100 * rules.sub_hundred.sh);</tag>
            hundred
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty.tens;</tag></item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out += rules.teens.teen;</
tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits.num;</
tag></item>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>
</grammar>
```

Date d'expiration de carte de crédit

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="dateCardExpiration"
         mode="voice"
         tag-format="semantics/1.0">
```

```
        <rule id="dateCardExpiration" scope="public">
            <tag>out=""</tag>
            <item repeat="1"><ruleref uri="#months"/><tag>out = out + rules.months;</
tag></item>
            <item repeat="1"><ruleref uri="#year"/><tag>out += " " + rules.year.yr;</
tag></item>
        </rule>

        <!-- Test Cases

        Grammar will support the following inputs:

            Scenario 1:
                Input: My card expiration date is july eleven
                Output: 07 2011

            Scenario 2:
                Input: My card expiration date is may twenty six
                Output: 05 2026

        -->

        <rule id="text">
           <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">My card expiration date is </item>
              <item repeat="0-1">Expiration date is </item>
           </one-of>
        </rule>

        <rule id="hesitation">
          <one-of>
             <item>Hmm</item>
             <item>Mmm</item>
             <item>My</item>
          </one-of>
        </rule>

        <rule id="months">
           <item repeat="0-1"><ruleref uri="#text"/></item>
           <one-of>
               <item>january<tag>out="01";</tag></item>
               <item>february<tag>out="02";</tag></item>
               <item>march<tag>out="03";</tag></item>
```

```
            <item>april<tag>out="04";</tag></item>
            <item>may<tag>out="05";</tag></item>
            <item>june<tag>out="06";</tag></item>
            <item>july<tag>out="07";</tag></item>
            <item>august<tag>out="08";</tag></item>
            <item>september<tag>out="09";</tag></item>
            <item>october<tag>out="10";</tag></item>
            <item>november<tag>out="11";</tag></item>
            <item>december<tag>out="12";</tag></item>
            <item>jan<tag>out="01";</tag></item>
            <item>feb<tag>out="02";</tag></item>
            <item>aug<tag>out="08";</tag></item>
            <item>sept<tag>out="09";</tag></item>
            <item>oct<tag>out="10";</tag></item>
            <item>nov<tag>out="11";</tag></item>
            <item>dec<tag>out="12";</tag></item>
            <item>1<tag>out="01";</tag></item>
            <item>2<tag>out="02";</tag></item>
            <item>3<tag>out="03";</tag></item>
            <item>4<tag>out="04";</tag></item>
            <item>5<tag>out="05";</tag></item>
            <item>6<tag>out="06";</tag></item>
            <item>7<tag>out="07";</tag></item>
            <item>8<tag>out="08";</tag></item>
            <item>9<tag>out="09";</tag></item>
            <item>ten<tag>out="10";</tag></item>
            <item>eleven<tag>out="11";</tag></item>
            <item>twelve<tag>out="12";</tag></item>
        </one-of>
    </rule>

     <rule id="digits">
         <item repeat="0-1"><ruleref uri="#text"/></item>
         <one-of>
             <item>0<tag>out=0;</tag></item>
             <item>1<tag>out=1;</tag></item>
             <item>2<tag>out=2;</tag></item>
             <item>3<tag>out=3;</tag></item>
             <item>4<tag>out=4;</tag></item>
             <item>5<tag>out=5;</tag></item>
             <item>6<tag>out=6;</tag></item>
             <item>7<tag>out=7;</tag></item>
             <item>8<tag>out=8;</tag></item>
             <item>9<tag>out=9;</tag></item>
```

```
                <item>one<tag>out=1;</tag></item>
                <item>two<tag>out=2;</tag></item>
                <item>three<tag>out=3;</tag></item>
                <item>four<tag>out=4;</tag></item>
                <item>five<tag>out=5;</tag></item>
                <item>six<tag>out=6;</tag></item>
                <item>seven<tag>out=7;</tag></item>
                <item>eight<tag>out=8;</tag></item>
                <item>nine<tag>out=9;</tag></item>
            </one-of>
        </rule>

        <rule id="year">
          <tag>out.yr="20"</tag>
          <one-of>
              <item><ruleref uri="#teens"/><tag>out.yr += rules.teens;</tag></item>
              <item><ruleref uri="#above_twenty"/><tag>out.yr += rules.above_twenty;</
tag></item>
          </one-of>
        </rule>

        <rule id="teens">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>ten<tag>out=10;</tag></item>
                <item>eleven<tag>out=11;</tag></item>
                <item>twelve<tag>out=12;</tag></item>
                <item>thirteen<tag>out=13;</tag></item>
                <item>fourteen<tag>out=14;</tag></item>
                <item>fifteen<tag>out=15;</tag></item>
                <item>sixteen<tag>out=16;</tag></item>
                <item>seventeen<tag>out=17;</tag></item>
                <item>eighteen<tag>out=18;</tag></item>
                <item>nineteen<tag>out=19;</tag></item>
                <item>10<tag>out=10;</tag></item>
                <item>11<tag>out=11;</tag></item>
                <item>12<tag>out=12;</tag></item>
                <item>13<tag>out=13;</tag></item>
                <item>14<tag>out=14;</tag></item>
                <item>15<tag>out=15;</tag></item>
                <item>16<tag>out=16;</tag></item>
                <item>17<tag>out=17;</tag></item>
                <item>18<tag>out=18;</tag></item>
                <item>19<tag>out=19;</tag></item>
```

```
            </one-of>
        </rule>

        <rule id="above_twenty">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
                <item>forty<tag>out=40;</tag></item>
                <item>fifty<tag>out=50;</tag></item>
                <item>sixty<tag>out=60;</tag></item>
                <item>seventy<tag>out=70;</tag></item>
                <item>eighty<tag>out=80;</tag></item>
                <item>ninety<tag>out=90;</tag></item>
                <item>20<tag>out=20;</tag></item>
                <item>30<tag>out=30;</tag></item>
                <item>40<tag>out=40;</tag></item>
                <item>50<tag>out=50;</tag></item>
                <item>60<tag>out=60;</tag></item>
                <item>70<tag>out=70;</tag></item>
                <item>80<tag>out=80;</tag></item>
                <item>90<tag>out=90;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
</grammar>
```

Date du relevé

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:
```

```
            Scenario 1:
                Input: Show me statements from July Five Two Thousand and Eleven
                Output: 07/5/11

            Scenario 2:
                Input: Show me statements from July Sixteen Two Thousand and Eleven
                Output: 07/16/11

            Scenario 3:
                Input: Show me statements from July Thirty Two Thousand and Eleven
                Output: 07/30/11
        -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item>
                <item repeat="1"><ruleref uri="#months"/><tag>out = out +
 rules.months.mon + "/";</tag></item>
                <one-of>
                  <item><ruleref uri="#digits"/><tag>out += rules.digits + "/";</
tag></item>
                  <item><ruleref uri="#teens"/><tag>out += rules.teens+ "/";</tag></
item>
                  <item><ruleref uri="#above_twenty"/><tag>out += rules.above_twenty+
 "/";</tag></item>
                </one-of>
                <one-of>
                    <item><ruleref uri="#thousands"/><tag>out += rules.thousands;</
tag></item>
                    <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
 rules.digits;</tag></item>
                    <item repeat="0-1"><ruleref uri="#teens"/><tag>out +=
 rules.teens;</tag></item>
                    <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty;</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="text">
          <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">I want to see bank statements from </item>
```

```
            <item repeat="0-1">Show me statements from</item>
         </one-of>
      </rule>

      <rule id="hesitation">
         <one-of>
            <item>Hmm</item>
            <item>Mmm</item>
            <item>My</item>
         </one-of>
       </rule>

      <rule id="months">
         <tag>out.mon=""</tag>
      <item repeat="0-1"><ruleref uri="#text"/></item>
         <one-of>
           <item>january<tag>out.mon+="01";</tag></item>
           <item>february<tag>out.mon+="02";</tag></item>
           <item>march<tag>out.mon+="03";</tag></item>
           <item>april<tag>out.mon+="04";</tag></item>
           <item>may<tag>out.mon+="05";</tag></item>
           <item>june<tag>out.mon+="06";</tag></item>
           <item>july<tag>out.mon+="07";</tag></item>
           <item>august<tag>out.mon+="08";</tag></item>
           <item>september<tag>out.mon+="09";</tag></item>
           <item>october<tag>out.mon+="10";</tag></item>
           <item>november<tag>out.mon+="11";</tag></item>
           <item>december<tag>out.mon+="12";</tag></item>
           <item>jan<tag>out.mon+="01";</tag></item>
           <item>feb<tag>out.mon+="02";</tag></item>
           <item>aug<tag>out.mon+="08";</tag></item>
           <item>sept<tag>out.mon+="09";</tag></item>
           <item>oct<tag>out.mon+="10";</tag></item>
           <item>nov<tag>out.mon+="11";</tag></item>
           <item>dec<tag>out.mon+="12";</tag></item>
         </one-of>
      </rule>

       <rule id="digits">
           <one-of>
               <item>zero<tag>out=0;</tag></item>
               <item>one<tag>out=1;</tag></item>
               <item>two<tag>out=2;</tag></item>
               <item>three<tag>out=3;</tag></item>
```

```
                <item>four<tag>out=4;</tag></item>
                <item>five<tag>out=5;</tag></item>
                <item>six<tag>out=6;</tag></item>
                <item>seven<tag>out=7;</tag></item>
                <item>eight<tag>out=8;</tag></item>
                <item>nine<tag>out=9;</tag></item>
            </one-of>
        </rule>

        <rule id="teens">
            <one-of>
                <item>ten<tag>out=10;</tag></item>
                <item>eleven<tag>out=11;</tag></item>
                <item>twelve<tag>out=12;</tag></item>
                <item>thirteen<tag>out=13;</tag></item>
                <item>fourteen<tag>out=14;</tag></item>
                <item>fifteen<tag>out=15;</tag></item>
                <item>sixteen<tag>out=16;</tag></item>
                <item>seventeen<tag>out=17;</tag></item>
                <item>eighteen<tag>out=18;</tag></item>
                <item>nineteen<tag>out=19;</tag></item>
            </one-of>
        </rule>

        <rule id="thousands">
            <item>two thousand</item>
            <item repeat="0-1">and</item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out = rules.digits;</
tag></item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out = rules.teens;</tag></
item>
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out =
 rules.above_twenty;</tag></item>
        </rule>

        <rule id="above_twenty">
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
```

```
 </grammar>
```

Date de transaction

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: My last incorrect transaction date is july twenty three
                 Output: 07/23

             Scenario 2:
                 Input: My last incorrect transaction date is july fifteen
                 Output: 07/15

         -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item repeat="1-10">
                <item><ruleref uri="#months"/><tag>out= rules.months.mon + "/";</tag></
item>
                <one-of>
                    <item><ruleref uri="#digits"/><tag>out+= rules.digits;</tag></item>
                    <item><ruleref uri="#teens"/><tag>out+= rules.teens;</tag></item>
                    <item><ruleref uri="#above_twenty"/><tag>out+=
 rules.above_twenty;</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="text">
```

```
        <item repeat="0-1"><ruleref uri="#hesitation"/></item>
      <one-of>
         <item repeat="0-1">My last incorrect transaction date is</item>
         <item repeat="0-1">It is</item>
      </one-of>
   </rule>
   <rule id="hesitation">
       <one-of>
          <item>Hmm</item>
          <item>Mmm</item>
          <item>My</item>
       </one-of>
     </rule>

   <rule id="months">
      <item repeat="0-1"><ruleref uri="#text"/></item>
      <tag>out.mon=""</tag>
      <one-of>
          <item>january<tag>out.mon+="01";</tag></item>
          <item>february<tag>out.mon+="02";</tag></item>
          <item>march<tag>out.mon+="03";</tag></item>
          <item>april<tag>out.mon+="04";</tag></item>
          <item>may<tag>out.mon+="05";</tag></item>
          <item>june<tag>out.mon+="06";</tag></item>
          <item>july<tag>out.mon+="07";</tag></item>
          <item>august<tag>out.mon+="08";</tag></item>
          <item>september<tag>out.mon+="09";</tag></item>
          <item>october<tag>out.mon+="10";</tag></item>
          <item>november<tag>out.mon+="11";</tag></item>
          <item>december<tag>out.mon+="12";</tag></item>
          <item>jan<tag>out.mon+="01";</tag></item>
          <item>feb<tag>out.mon+="02";</tag></item>
          <item>aug<tag>out.mon+="08";</tag></item>
          <item>sept<tag>out.mon+="09";</tag></item>
          <item>oct<tag>out.mon+="10";</tag></item>
          <item>nov<tag>out.mon+="11";</tag></item>
          <item>dec<tag>out.mon+="12";</tag></item>
      </one-of>
   </rule>

   <rule id="digits">
       <item repeat="0-1"><ruleref uri="#text"/></item>
       <one-of>
           <item>0<tag>out=0;</tag></item>
```

```
            <item>1<tag>out=1;</tag></item>
            <item>2<tag>out=2;</tag></item>
            <item>3<tag>out=3;</tag></item>
            <item>4<tag>out=4;</tag></item>
            <item>5<tag>out=5;</tag></item>
            <item>6<tag>out=6;</tag></item>
            <item>7<tag>out=7;</tag></item>
            <item>8<tag>out=8;</tag></item>
            <item>9<tag>out=9;</tag></item>
            <item>first<tag>out=01;</tag></item>
            <item>second<tag>out=02;</tag></item>
            <item>third<tag>out=03;</tag></item>
            <item>fourth<tag>out=04;</tag></item>
            <item>fifth<tag>out=05;</tag></item>
            <item>sixth<tag>out=06;</tag></item>
            <item>seventh<tag>out=07;</tag></item>
            <item>eighth<tag>out=08;</tag></item>
            <item>ninth<tag>out=09;</tag></item>
            <item>one<tag>out=1;</tag></item>
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>

    <rule id="teens">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>ten<tag>out=10;</tag></item>
            <item>tenth<tag>out=10;</tag></item>
            <item>eleven<tag>out=11;</tag></item>
            <item>twelve<tag>out=12;</tag></item>
            <item>thirteen<tag>out=13;</tag></item>
            <item>fourteen<tag>out=14;</tag></item>
            <item>fifteen<tag>out=15;</tag></item>
            <item>sixteen<tag>out=16;</tag></item>
            <item>seventeen<tag>out=17;</tag></item>
            <item>eighteen<tag>out=18;</tag></item>
            <item>nineteen<tag>out=19;</tag></item>
```

```
                <item>tenth<tag>out=10;</tag></item>
                <item>eleventh<tag>out=11;</tag></item>
                <item>twelveth<tag>out=12;</tag></item>
                <item>thirteenth<tag>out=13;</tag></item>
                <item>fourteenth<tag>out=14;</tag></item>
                <item>fifteenth<tag>out=15;</tag></item>
                <item>sixteenth<tag>out=16;</tag></item>
                <item>seventeenth<tag>out=17;</tag></item>
                <item>eighteenth<tag>out=18;</tag></item>
                <item>nineteenth<tag>out=19;</tag></item>
            </one-of>
        </rule>

        <rule id="above_twenty">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
</grammar>
```

Montant du transfert

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: I want to transfer twenty five Dollar
                 Output: $25
```

```
             Scenario 2:
                 Input: transfer twenty five dollars
                 Output: $25

         -->

        <rule id="main" scope="public">
            <tag>out="$"</tag>
            <one-of>
                <item><ruleref uri="#sub_hundred"/><tag>out += rules.sub_hundred.sh;</
tag></item>
                <item><ruleref uri="#subThousands"/><tag>out += rules.subThousands;</
tag></item>
            </one-of>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">I want to transfer</item>
              <item repeat="0-1">transfer</item>
              <item repeat="0-1">make a transfer for</item>
           </one-of>
        </rule>

        <rule id="hesitation">
            <one-of>
               <item>Hmm</item>
               <item>Mmm</item>
               <item>My</item>
            </one-of>
          </rule>

        <rule id="digits">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.num = 0;</tag>
            <one-of>
                <item>0<tag>out.num+=0;</tag></item>
                <item>1<tag>out.num+=1;</tag></item>
                <item>2<tag>out.num+=2;</tag></item>
                <item>3<tag>out.num+=3;</tag></item>
                <item>4<tag>out.num+=4;</tag></item>
                <item>5<tag>out.num+=5;</tag></item>
```

```
            <item>6<tag>out.num+=6;</tag></item>
            <item>7<tag>out.num+=7;</tag></item>
            <item>8<tag>out.num+=8;</tag></item>
            <item>9<tag>out.num+=9;</tag></item>
            <item>one<tag>out.num+=1;</tag></item>
            <item>two<tag>out.num+=2;</tag></item>
            <item>three<tag>out.num+=3;</tag></item>
            <item>four<tag>out.num+=4;</tag></item>
            <item>five<tag>out.num+=5;</tag></item>
            <item>six<tag>out.num+=6;</tag></item>
            <item>seven<tag>out.num+=7;</tag></item>
            <item>eight<tag>out.num+=8;</tag></item>
            <item>nine<tag>out.num+=9;</tag></item>
        </one-of>
        <item repeat="0-1"><ruleref uri="#currency"/></item>
    </rule>

    <rule id="teens">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <tag>out.teen = 0;</tag>
        <one-of>
            <item>ten<tag>out.teen+=10;</tag></item>
            <item>eleven<tag>out.teen+=11;</tag></item>
            <item>twelve<tag>out.teen+=12;</tag></item>
            <item>thirteen<tag>out.teen+=13;</tag></item>
            <item>fourteen<tag>out.teen+=14;</tag></item>
            <item>fifteen<tag>out.teen+=15;</tag></item>
            <item>sixteen<tag>out.teen+=16;</tag></item>
            <item>seventeen<tag>out.teen+=17;</tag></item>
            <item>eighteen<tag>out.teen+=18;</tag></item>
            <item>nineteen<tag>out.teen+=19;</tag></item>
        </one-of>
        <item repeat="0-1"><ruleref uri="#currency"/></item>
    </rule>

    <rule id="above_twenty">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <tag>out.tens = 0;</tag>
        <one-of>
            <item>twenty<tag>out.tens+=20;</tag></item>
            <item>thirty<tag>out.tens+=30;</tag></item>
            <item>forty<tag>out.tens+=40;</tag></item>
            <item>fifty<tag>out.tens+=50;</tag></item>
            <item>sixty<tag>out.tens+=60;</tag></item>
```

```
                <item>seventy<tag>out.tens+=70;</tag></item>
                <item>eighty<tag>out.tens+=80;</tag></item>
                <item>ninety<tag>out.tens+=90;</tag></item>
                <item>hundred<tag>out.tens+=100;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.tens +=
 rules.digits.num;</tag></item>
        </rule>

        <rule id="currency">
            <one-of>
                <item repeat="0-1">dollars</item>
                <item repeat="0-1">Dollars</item>
                <item repeat="0-1">dollar</item>
                <item repeat="0-1">Dollar</item>
            </one-of>
        </rule>


        <rule id="sub_hundred">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.sh = 0;</tag>
            <one-of>
                <item><ruleref uri="#teens"/><tag>out.sh += rules.teens.teen;</tag></
item>
                <item>
                    <ruleref uri="#above_twenty"/><tag>out.sh +=
 rules.above_twenty.tens;</tag>
                </item>
                <item><ruleref uri="#digits"/><tag>out.sh += rules.digits.num;</tag></
item>
            </one-of>
         </rule>

        <rule id="subThousands">
            <ruleref uri="#sub_hundred"/><tag>out = (100 * rules.sub_hundred.sh);</tag>
            hundred
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty.tens;</tag></item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out += rules.teens.teen;</
tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits.num;</
tag></item>
```

```
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>
</grammar>
```

Numéro de sécurité sociale

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <rule id="main" scope="public">
             <tag>out=""</tag>
             <ruleref uri="#digits"/><tag>out += rules.digits.numbers;</tag>
         </rule>

         <rule id="digits">
             <tag>out.numbers=""</tag>
             <item repeat="1-12">
                 <one-of>
                     <item>0<tag>out.numbers+=0;</tag></item>
                     <item>1<tag>out.numbers+=1;</tag></item>
                     <item>2<tag>out.numbers+=2;</tag></item>
                     <item>3<tag>out.numbers+=3;</tag></item>
                     <item>4<tag>out.numbers+=4;</tag></item>
                     <item>5<tag>out.numbers+=5;</tag></item>
                     <item>6<tag>out.numbers+=6;</tag></item>
                     <item>7<tag>out.numbers+=7;</tag></item>
                     <item>8<tag>out.numbers+=8;</tag></item>
                     <item>9<tag>out.numbers+=9;</tag></item>
                     <item>zero<tag>out.numbers+=0;</tag></item>
                     <item>one<tag>out.numbers+=1;</tag></item>
                     <item>two<tag>out.numbers+=2;</tag></item>
                     <item>three<tag>out.numbers+=3;</tag></item>
                     <item>four<tag>out.numbers+=4;</tag></item>
                     <item>five<tag>out.numbers+=5;</tag></item>
                     <item>six<tag>out.numbers+=6;</tag></item>
                     <item>seven<tag>out.numbers+=7;</tag></item>
```

```
                    <item>eight<tag>out.numbers+=8;</tag></item>
                    <item>nine<tag>out.numbers+=9;</tag></item>
                    <item>dash</item>
                </one-of>
            </item>
        </rule>
</grammar>
```

Grammaires pour les assurances (téléchargement)

Les grammaires suivantes sont prises en charge pour le domaine de l'assurance : numéros de réclamation et de police, numéros de permis de conduire et de plaque d'immatriculation, dates d'expiration, dates de début et dates de renouvellement, montants des sinistres et des polices.

ID de réclamation

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="digits"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: My claim number is One Five Four Two
                 Output: 1542

             Scenario 2:
                 Input: Claim number One Five Four Four
                 Output: 1544

         -->

        <rule id="digits">
            <tag>out=""</tag>
```

```
            <item><ruleref uri="#singleDigit"/><tag>out += rules.singleDigit.digit;</
tag></item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
            <one-of>
                <item repeat="0-1">My claim number is</item>
                <item repeat="0-1">Claim number</item>
                <item repeat="0-1">This is for claim</item>
            </one-of>
        </rule>

        <rule id="hesitation">
            <one-of>
               <item>Hmm</item>
               <item>Mmm</item>
               <item>My</item>
            </one-of>
          </rule>

        <rule id="singleDigit">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.digit=""</tag>
            <item repeat="1-10">
                <one-of>
                    <item>0<tag>out.digit+=0;</tag></item>
                    <item>zero<tag>out.digit+=0;</tag></item>
                    <item>1<tag>out.digit+=1;</tag></item>
                    <item>one<tag>out.digit+=1;</tag></item>
                    <item>2<tag>out.digit+=2;</tag></item>
                    <item>two<tag>out.digit+=2;</tag></item>
                    <item>3<tag>out.digit+=3;</tag></item>
                    <item>three<tag>out.digit+=3;</tag></item>
                    <item>4<tag>out.digit+=4;</tag></item>
                    <item>four<tag>out.digit+=4;</tag></item>
                    <item>5<tag>out.digit+=5;</tag></item>
                    <item>five<tag>out.digit+=5;</tag></item>
                    <item>6<tag>out.digit+=6;</tag></item>
                    <item>six<tag>out.digit+=5;</tag></item>
                    <item>7<tag>out.digit+=7;</tag></item>
                    <item>seven<tag>out.digit+=7;</tag></item>
                    <item>8<tag>out.digit+=8;</tag></item>
                    <item>eight<tag>out.digit+=8;</tag></item>
```

```
                    <item>9<tag>out.digit+=9;</tag></item>
                    <item>nine<tag>out.digit+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

ID de stratégie

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">


        <!-- Test Cases

        Grammar will support the following inputs:

            Scenario 1:
                Input: My policy number is A B C 1 2 3 4
                Output: ABC1234

            Scenario 2:
                Input: This is the policy number 1 2 3 4 A B C
                Output: 1234ABC

            Scenario 3:
                Input: Hmm My policy number is 1 2 3 4 A B C 1
                Output: 123ABC1
        -->

        <rule id="main" scope="public">
            <tag>out="";</tag>
            <item><ruleref uri="#alphanumeric"/><tag>out +=
 rules.alphanumeric.alphanum;</tag></item>
            <item repeat="0-1"><ruleref uri="#alphabets"/><tag>out +=
 rules.alphabets.letters;</tag></item>
```

```
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
rules.digits.numbers</tag></item>
            <item repeat="0-1"><ruleref uri="#thanks"/></item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
            <one-of>
                <item repeat="0-1">My policy number is</item>
                <item repeat="0-1">This is the policy number</item>
                <item repeat="0-1">Policy number</item>
                <item repeat="0-1">Yes, It is</item>
                <item repeat="0-1">Yes It is</item>
                <item repeat="0-1">Yes It's</item>
                <item repeat="0-1">My policy Id is</item>
                <item repeat="0-1">This is the policy Id</item>
                <item repeat="0-1">Policy Id</item>
            </one-of>
        </rule>

        <rule id="hesitation">
          <one-of>
             <item>Hmm</item>
             <item>Mmm</item>
             <item>My</item>
          </one-of>
        </rule>

        <rule id="thanks">
            <one-of>
               <item>Thanks</item>
               <item>I think</item>
            </one-of>
          </rule>

        <rule id="alphanumeric" scope="public">
            <tag>out.alphanum=""</tag>
            <item><ruleref uri="#alphabets"/><tag>out.alphanum +=
rules.alphabets.letters;</tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.alphanum +=
rules.digits.numbers</tag></item>
        </rule>

        <rule id="alphabets">
```

```
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.letters=""</tag>
            <tag>out.firstOccurence=""</tag>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.firstOccurence +=
rules.digits.numbers; out.letters += out.firstOccurence;</tag></item>
            <item repeat="1-">
                <one-of>
                    <item>A<tag>out.letters+='A';</tag></item>
                    <item>B<tag>out.letters+='B';</tag></item>
                    <item>C<tag>out.letters+='C';</tag></item>
                    <item>D<tag>out.letters+='D';</tag></item>
                    <item>E<tag>out.letters+='E';</tag></item>
                    <item>F<tag>out.letters+='F';</tag></item>
                    <item>G<tag>out.letters+='G';</tag></item>
                    <item>H<tag>out.letters+='H';</tag></item>
                    <item>I<tag>out.letters+='I';</tag></item>
                    <item>J<tag>out.letters+='J';</tag></item>
                    <item>K<tag>out.letters+='K';</tag></item>
                    <item>L<tag>out.letters+='L';</tag></item>
                    <item>M<tag>out.letters+='M';</tag></item>
                    <item>N<tag>out.letters+='N';</tag></item>
                    <item>O<tag>out.letters+='O';</tag></item>
                    <item>P<tag>out.letters+='P';</tag></item>
                    <item>Q<tag>out.letters+='Q';</tag></item>
                    <item>R<tag>out.letters+='R';</tag></item>
                    <item>S<tag>out.letters+='S';</tag></item>
                    <item>T<tag>out.letters+='T';</tag></item>
                    <item>U<tag>out.letters+='U';</tag></item>
                    <item>V<tag>out.letters+='V';</tag></item>
                    <item>W<tag>out.letters+='W';</tag></item>
                    <item>X<tag>out.letters+='X';</tag></item>
                    <item>Y<tag>out.letters+='Y';</tag></item>
                    <item>Z<tag>out.letters+='Z';</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="digits">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.numbers=""</tag>
            <item repeat="1-10">
                <one-of>
                    <item>0<tag>out.numbers+=0;</tag></item>
                    <item>1<tag>out.numbers+=1;</tag></item>
```

```
                    <item>2<tag>out.numbers+=2;</tag></item>
                    <item>3<tag>out.numbers+=3;</tag></item>
                    <item>4<tag>out.numbers+=4;</tag></item>
                    <item>5<tag>out.numbers+=5;</tag></item>
                    <item>6<tag>out.numbers+=6;</tag></item>
                    <item>7<tag>out.numbers+=7;</tag></item>
                    <item>8<tag>out.numbers+=8;</tag></item>
                    <item>9<tag>out.numbers+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

Numéro du permis de conduire

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="digits"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: My drivers license number is One Five Four Two
                 Output: 1542

             Scenario 2:
                 Input: driver license number One Five Four Four
                 Output: 1544

         -->

        <rule id="digits">
            <tag>out=""</tag>
            <item><ruleref uri="#singleDigit"/><tag>out += rules.singleDigit.digit;</
tag></item>
```

```
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
            <one-of>
                <item repeat="0-1">My drivers license number is</item>
                <item repeat="0-1">My drivers license id is</item>
                <item repeat="0-1">Driver license number</item>
            </one-of>
        </rule>

        <rule id="hesitation">
            <one-of>
               <item>Hmm</item>
               <item>Mmm</item>
               <item>My</item>
            </one-of>
          </rule>

        <rule id="singleDigit">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.digit=""</tag>
            <item repeat="1-10">
                <one-of>
                    <item>0<tag>out.digit+=0;</tag></item>
                    <item>zero<tag>out.digit+=0;</tag></item>
                    <item>1<tag>out.digit+=1;</tag></item>
                    <item>one<tag>out.digit+=1;</tag></item>
                    <item>2<tag>out.digit+=2;</tag></item>
                    <item>two<tag>out.digit+=2;</tag></item>
                    <item>3<tag>out.digit+=3;</tag></item>
                    <item>three<tag>out.digit+=3;</tag></item>
                    <item>4<tag>out.digit+=4;</tag></item>
                    <item>four<tag>out.digit+=4;</tag></item>
                    <item>5<tag>out.digit+=5;</tag></item>
                    <item>five<tag>out.digit+=5;</tag></item>
                    <item>6<tag>out.digit+=6;</tag></item>
                    <item>six<tag>out.digit+=5;</tag></item>
                    <item>7<tag>out.digit+=7;</tag></item>
                    <item>seven<tag>out.digit+=7;</tag></item>
                    <item>8<tag>out.digit+=8;</tag></item>
                    <item>eight<tag>out.digit+=8;</tag></item>
                    <item>9<tag>out.digit+=9;</tag></item>
                    <item>nine<tag>out.digit+=9;</tag></item>
```

```
                </one-of>
            </item>
        </rule>
</grammar>
```

Numéro de plaque d'immatriculation

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">


        <!-- Test Cases

        Grammar will support the following inputs:

            Scenario 1:
                Input: my license plate is A B C D 1 2
                Output: ABCD12

            Scenario 2:
                Input: license plate number A B C 1 2 3 4
                Output: ABC1234

            Scenario 3:
                Input: my plates say A F G K 9 8 7 6 Thanks
                Output: AFGK9876
        -->

        <rule id="main" scope="public">
            <tag>out.licenseNum=""</tag>
            <item><ruleref uri="#alphabets"/><tag>out.licenseNum +=
 rules.alphabets.letters;</tag></item>
            <item repeat="0-1"><ruleref uri="#thanks"/></item>
        </rule>

        <rule id="text">
```

```
        <item repeat="0-1"><ruleref uri="#hesitation"/></item>
        <one-of>
            <item repeat="0-1">my license plate is</item>
            <item repeat="0-1">license plate number</item>
            <item repeat="0-1">my plates say</item>
        </one-of>
    </rule>

    <rule id="hesitation">
      <one-of>
         <item>Hmm</item>
         <item>Mmm</item>
         <item>My</item>
      </one-of>
    </rule>

    <rule id="thanks">
        <one-of>
           <item>Thanks</item>
           <item>I think</item>
        </one-of>
      </rule>

    <rule id="alphabets">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <tag>out.letters=""</tag>
        <tag>out.firstOccurence=""</tag>
        <item repeat="3-4">
            <one-of>
                <item>A<tag>out.letters+='A';</tag></item>
                <item>B<tag>out.letters+='B';</tag></item>
                <item>C<tag>out.letters+='C';</tag></item>
                <item>D<tag>out.letters+='D';</tag></item>
                <item>E<tag>out.letters+='E';</tag></item>
                <item>F<tag>out.letters+='F';</tag></item>
                <item>G<tag>out.letters+='G';</tag></item>
                <item>H<tag>out.letters+='H';</tag></item>
                <item>I<tag>out.letters+='I';</tag></item>
                <item>J<tag>out.letters+='J';</tag></item>
                <item>K<tag>out.letters+='K';</tag></item>
                <item>L<tag>out.letters+='L';</tag></item>
                <item>M<tag>out.letters+='M';</tag></item>
                <item>N<tag>out.letters+='N';</tag></item>
                <item>O<tag>out.letters+='O';</tag></item>
```

```
                    <item>P<tag>out.letters+='P';</tag></item>
                    <item>Q<tag>out.letters+='Q';</tag></item>
                    <item>R<tag>out.letters+='R';</tag></item>
                    <item>S<tag>out.letters+='S';</tag></item>
                    <item>T<tag>out.letters+='T';</tag></item>
                    <item>U<tag>out.letters+='U';</tag></item>
                    <item>V<tag>out.letters+='V';</tag></item>
                    <item>W<tag>out.letters+='W';</tag></item>
                    <item>X<tag>out.letters+='X';</tag></item>
                    <item>Y<tag>out.letters+='Y';</tag></item>
                    <item>Z<tag>out.letters+='Z';</tag></item>
                </one-of>
            </item>
    <item repeat="0-1"><ruleref uri="#digits"/><tag>out.firstOccurence +=
 rules.digits.numbers; out.letters += out.firstOccurence;</tag></item>
        </rule>

        <rule id="digits">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.numbers=""</tag>
            <item repeat="2-4">
                <one-of>
                    <item>0<tag>out.numbers+=0;</tag></item>
                    <item>1<tag>out.numbers+=1;</tag></item>
                    <item>2<tag>out.numbers+=2;</tag></item>
                    <item>3<tag>out.numbers+=3;</tag></item>
                    <item>4<tag>out.numbers+=4;</tag></item>
                    <item>5<tag>out.numbers+=5;</tag></item>
                    <item>6<tag>out.numbers+=6;</tag></item>
                    <item>7<tag>out.numbers+=7;</tag></item>
                    <item>8<tag>out.numbers+=8;</tag></item>
                    <item>9<tag>out.numbers+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

Date d'expiration de carte de crédit

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
```

```
                              http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="dateCardExpiration"
         mode="voice"
         tag-format="semantics/1.0">

        <rule id="dateCardExpiration" scope="public">
            <tag>out=""</tag>
            <item repeat="1"><ruleref uri="#months"/><tag>out = out + rules.months;</
tag></item>
            <item repeat="1"><ruleref uri="#year"/><tag>out += " " + rules.year.yr;</
tag></item>
            <item repeat="0-1"><ruleref uri="#thanks"/></item>
        </rule>

        <!-- Test Cases

        Grammar will support the following inputs:

            Scenario 1:
                Input: My card expiration date is july eleven
                Output: 07 2011

            Scenario 2:
                Input: My card expiration date is may twenty six
                Output: 05 2026

        -->

        <rule id="text">
           <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">My card expiration date is </item>
           </one-of>
        </rule>

        <rule id="hesitation">
          <one-of>
             <item>Hmm</item>
             <item>Mmm</item>
             <item>My</item>
          </one-of>
        </rule>
```

```
        <rule id="thanks">
            <one-of>
               <item>Thanks</item>
               <item>I think</item>
            </one-of>
          </rule>

        <rule id="months">
           <item repeat="0-1"><ruleref uri="#text"/></item>
           <one-of>
               <item>january<tag>out="01";</tag></item>
               <item>february<tag>out="02";</tag></item>
               <item>march<tag>out="03";</tag></item>
               <item>april<tag>out="04";</tag></item>
               <item>may<tag>out="05";</tag></item>
               <item>june<tag>out="06";</tag></item>
               <item>july<tag>out="07";</tag></item>
               <item>august<tag>out="08";</tag></item>
               <item>september<tag>out="09";</tag></item>
               <item>october<tag>out="10";</tag></item>
               <item>november<tag>out="11";</tag></item>
               <item>december<tag>out="12";</tag></item>
               <item>jan<tag>out="01";</tag></item>
               <item>feb<tag>out="02";</tag></item>
               <item>aug<tag>out="08";</tag></item>
               <item>sept<tag>out="09";</tag></item>
               <item>oct<tag>out="10";</tag></item>
               <item>nov<tag>out="11";</tag></item>
               <item>dec<tag>out="12";</tag></item>
               <item>1<tag>out="01";</tag></item>
               <item>2<tag>out="02";</tag></item>
               <item>3<tag>out="03";</tag></item>
               <item>4<tag>out="04";</tag></item>
               <item>5<tag>out="05";</tag></item>
               <item>6<tag>out="06";</tag></item>
               <item>7<tag>out="07";</tag></item>
               <item>8<tag>out="08";</tag></item>
               <item>9<tag>out="09";</tag></item>
               <item>ten<tag>out="10";</tag></item>
               <item>eleven<tag>out="11";</tag></item>
               <item>twelve<tag>out="12";</tag></item>
           </one-of>
       </rule>
```

```
        <rule id="digits">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>0<tag>out=0;</tag></item>
                <item>1<tag>out=1;</tag></item>
                <item>2<tag>out=2;</tag></item>
                <item>3<tag>out=3;</tag></item>
                <item>4<tag>out=4;</tag></item>
                <item>5<tag>out=5;</tag></item>
                <item>6<tag>out=6;</tag></item>
                <item>7<tag>out=7;</tag></item>
                <item>8<tag>out=8;</tag></item>
                <item>9<tag>out=9;</tag></item>
                <item>one<tag>out=1;</tag></item>
                <item>two<tag>out=2;</tag></item>
                <item>three<tag>out=3;</tag></item>
                <item>four<tag>out=4;</tag></item>
                <item>five<tag>out=5;</tag></item>
                <item>six<tag>out=6;</tag></item>
                <item>seven<tag>out=7;</tag></item>
                <item>eight<tag>out=8;</tag></item>
                <item>nine<tag>out=9;</tag></item>
            </one-of>
        </rule>

        <rule id="year">
          <tag>out.yr="20"</tag>
          <one-of>
              <item><ruleref uri="#teens"/><tag>out.yr += rules.teens;</tag></item>
              <item><ruleref uri="#above_twenty"/><tag>out.yr += rules.above_twenty;</
tag></item>
          </one-of>
        </rule>

        <rule id="teens">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>ten<tag>out=10;</tag></item>
                <item>eleven<tag>out=11;</tag></item>
                <item>twelve<tag>out=12;</tag></item>
                <item>thirteen<tag>out=13;</tag></item>
                <item>fourteen<tag>out=14;</tag></item>
                <item>fifteen<tag>out=15;</tag></item>
                <item>sixteen<tag>out=16;</tag></item>
```

```
                <item>seventeen<tag>out=17;</tag></item>
                <item>eighteen<tag>out=18;</tag></item>
                <item>nineteen<tag>out=19;</tag></item>
                <item>10<tag>out=10;</tag></item>
                <item>11<tag>out=11;</tag></item>
                <item>12<tag>out=12;</tag></item>
                <item>13<tag>out=13;</tag></item>
                <item>14<tag>out=14;</tag></item>
                <item>15<tag>out=15;</tag></item>
                <item>16<tag>out=16;</tag></item>
                <item>17<tag>out=17;</tag></item>
                <item>18<tag>out=18;</tag></item>
                <item>19<tag>out=19;</tag></item>
            </one-of>
        </rule>

        <rule id="above_twenty">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
                <item>forty<tag>out=40;</tag></item>
                <item>fifty<tag>out=50;</tag></item>
                <item>sixty<tag>out=60;</tag></item>
                <item>seventy<tag>out=70;</tag></item>
                <item>eighty<tag>out=80;</tag></item>
                <item>ninety<tag>out=90;</tag></item>
                <item>20<tag>out=20;</tag></item>
                <item>30<tag>out=30;</tag></item>
                <item>40<tag>out=40;</tag></item>
                <item>50<tag>out=50;</tag></item>
                <item>60<tag>out=60;</tag></item>
                <item>70<tag>out=70;</tag></item>
                <item>80<tag>out=80;</tag></item>
                <item>90<tag>out=90;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
</grammar>
```

Date d'expiration du contrat, jour/mois/année

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: My policy expired on July Five Two Thousand and Eleven
                 Output: 07/5/11

             Scenario 2:
                 Input: My policy will expire on July Sixteen Two Thousand and Eleven
                 Output: 07/16/11

             Scenario 3:
                 Input: My policy expired on July Thirty Two Thousand and Eleven
                 Output: 07/30/11
         -->

         <rule id="main" scope="public">
             <tag>out=""</tag>
             <item>
                 <item repeat="1"><ruleref uri="#months"/><tag>out = out +
 rules.months.mon + "/";</tag></item>
                 <one-of>
                   <item><ruleref uri="#digits"/><tag>out += rules.digits + "/";</
tag></item>
                   <item><ruleref uri="#teens"/><tag>out += rules.teens+ "/";</tag></
item>
                   <item><ruleref uri="#above_twenty"/><tag>out += rules.above_twenty+
 "/";</tag></item>
                 </one-of>
                 <one-of>
```

```
                    <item><ruleref uri="#thousands"/><tag>out += rules.thousands;</
tag></item>
                    <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
 rules.digits;</tag></item>
                    <item repeat="0-1"><ruleref uri="#teens"/><tag>out +=
 rules.teens;</tag></item>
                    <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty;</tag></item>
                 </one-of>
             </item>
        </rule>

        <rule id="text">
          <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">My policy expired on</item>
              <item repeat="0-1">My policy will expire on</item>
           </one-of>
        </rule>

        <rule id="hesitation">
           <one-of>
              <item>Hmm</item>
              <item>Mmm</item>
              <item>My</item>
           </one-of>
         </rule>

        <rule id="months">
           <tag>out.mon=""</tag>
    <item repeat="0-1"><ruleref uri="#text"/></item>
           <one-of>
             <item>january<tag>out.mon+="01";</tag></item>
             <item>february<tag>out.mon+="02";</tag></item>
             <item>march<tag>out.mon+="03";</tag></item>
             <item>april<tag>out.mon+="04";</tag></item>
             <item>may<tag>out.mon+="05";</tag></item>
             <item>june<tag>out.mon+="06";</tag></item>
             <item>july<tag>out.mon+="07";</tag></item>
             <item>august<tag>out.mon+="08";</tag></item>
             <item>september<tag>out.mon+="09";</tag></item>
             <item>october<tag>out.mon+="10";</tag></item>
             <item>november<tag>out.mon+="11";</tag></item>
             <item>december<tag>out.mon+="12";</tag></item>
```

```
            <item>jan<tag>out.mon+="01";</tag></item>
            <item>feb<tag>out.mon+="02";</tag></item>
            <item>aug<tag>out.mon+="08";</tag></item>
            <item>sept<tag>out.mon+="09";</tag></item>
            <item>oct<tag>out.mon+="10";</tag></item>
            <item>nov<tag>out.mon+="11";</tag></item>
            <item>dec<tag>out.mon+="12";</tag></item>
        </one-of>
    </rule>

    <rule id="digits">
        <one-of>
            <item>zero<tag>out=0;</tag></item>
            <item>one<tag>out=1;</tag></item>
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>

    <rule id="teens">
        <one-of>
            <item>ten<tag>out=10;</tag></item>
            <item>eleven<tag>out=11;</tag></item>
            <item>twelve<tag>out=12;</tag></item>
            <item>thirteen<tag>out=13;</tag></item>
            <item>fourteen<tag>out=14;</tag></item>
            <item>fifteen<tag>out=15;</tag></item>
            <item>sixteen<tag>out=16;</tag></item>
            <item>seventeen<tag>out=17;</tag></item>
            <item>eighteen<tag>out=18;</tag></item>
            <item>nineteen<tag>out=19;</tag></item>
        </one-of>
    </rule>

    <rule id="thousands">
        <item>two thousand</item>
        <item repeat="0-1">and</item>
```

```
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out = rules.digits;</
tag></item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out = rules.teens;</tag></
item>
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out =
 rules.above_twenty;</tag></item>
        </rule>

        <rule id="above_twenty">
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
 </grammar>
```

Date de renouvellement du contrat, mois/année

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: I renewed my policy on July Two Thousand and Eleven
                 Output: 07/11

             Scenario 2:
                 Input: My policy will renew on July Two Thousand and Eleven
                 Output: 07/11

         -->
```

```
        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item repeat="1-10">
                <item repeat="1"><ruleref uri="#months"/><tag>out = out +
 rules.months.mon + "/";</tag></item>
                <one-of>
                    <item><ruleref uri="#thousands"/><tag>out += rules.thousands;</
tag></item>
                    <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
 rules.digits;</tag></item>
                    <item repeat="0-1"><ruleref uri="#teens"/><tag>out +=
 rules.teens;</tag></item>
                    <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty;</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
            <one-of>
                <item repeat="0-1">My policy will renew on</item>
                <item repeat="0-1">My policy was renewed on</item>
                <item repeat="0-1">Renew policy on</item>
                <item repeat="0-1">I renewed my policy on</item>
            </one-of>
        </rule>

        <rule id="hesitation">
            <one-of>
               <item>Hmm</item>
               <item>Mmm</item>
               <item>My</item>
            </one-of>
          </rule>

        <rule id="months">
           <item repeat="0-1"><ruleref uri="#text"/></item>
           <tag>out.mon=""</tag>
           <one-of>
               <item>january<tag>out.mon+="01";</tag></item>
               <item>february<tag>out.mon+="02";</tag></item>
               <item>march<tag>out.mon+="03";</tag></item>
```

```
            <item>april<tag>out.mon+="04";</tag></item>
            <item>may<tag>out.mon+="05";</tag></item>
            <item>june<tag>out.mon+="06";</tag></item>
            <item>july<tag>out.mon+="07";</tag></item>
            <item>august<tag>out.mon+="08";</tag></item>
            <item>september<tag>out.mon+="09";</tag></item>
            <item>october<tag>out.mon+="10";</tag></item>
            <item>november<tag>out.mon+="11";</tag></item>
            <item>december<tag>out.mon+="12";</tag></item>
            <item>jan<tag>out.mon+="01";</tag></item>
            <item>feb<tag>out.mon+="02";</tag></item>
            <item>aug<tag>out.mon+="08";</tag></item>
            <item>sept<tag>out.mon+="09";</tag></item>
            <item>oct<tag>out.mon+="10";</tag></item>
            <item>nov<tag>out.mon+="11";</tag></item>
            <item>dec<tag>out.mon+="12";</tag></item>
        </one-of>
    </rule>

    <rule id="digits">
        <one-of>
            <item>zero<tag>out=0;</tag></item>
            <item>one<tag>out=1;</tag></item>
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>

    <rule id="teens">
        <one-of>
            <item>ten<tag>out=10;</tag></item>
            <item>eleven<tag>out=11;</tag></item>
            <item>twelve<tag>out=12;</tag></item>
            <item>thirteen<tag>out=13;</tag></item>
            <item>fourteen<tag>out=14;</tag></item>
            <item>fifteen<tag>out=15;</tag></item>
            <item>sixteen<tag>out=16;</tag></item>
            <item>seventeen<tag>out=17;</tag></item>
```

```
                <item>eighteen<tag>out=18;</tag></item>
                <item>nineteen<tag>out=19;</tag></item>
            </one-of>
        </rule>

        <rule id="thousands">
            <item>two thousand<!--<tag>out=2000;</tag>--></item>
            <item repeat="0-1">and</item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out = rules.digits;</tag></
item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out = rules.teens;</tag></
item>
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out =
 rules.above_twenty;</tag></item>
        </rule>

        <rule id="above_twenty">
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
                <item>forty<tag>out=40;</tag></item>
                <item>fifty<tag>out=50;</tag></item>
                <item>sixty<tag>out=60;</tag></item>
                <item>seventy<tag>out=70;</tag></item>
                <item>eighty<tag>out=80;</tag></item>
                <item>ninety<tag>out=90;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
</grammar>
```

Date de début de la politique

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">
```

```
         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: I bought my policy on july twenty three
                 Output: 07/23

             Scenario 2:
                 Input: My policy started on july fifteen
                 Output: 07/15

         -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item repeat="1-10">
                <item><ruleref uri="#months"/><tag>out= rules.months.mon + "/";</tag></
item>
                <one-of>
                    <item><ruleref uri="#digits"/><tag>out+= rules.digits;</tag></item>
                    <item><ruleref uri="#teens"/><tag>out+= rules.teens;</tag></item>
                    <item><ruleref uri="#above_twenty"/><tag>out+=
 rules.above_twenty;</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
             <one-of>
                <item repeat="0-1">I bought my policy on</item>
                <item repeat="0-1">I bought policy on</item>
                <item repeat="0-1">My policy started on</item>
             </one-of>
          </rule>

          <rule id="hesitation">
             <one-of>
                <item>Hmm</item>
                <item>Mmm</item>
                <item>My</item>
             </one-of>
```

```
    </rule>

  <rule id="months">
    <item repeat="0-1"><ruleref uri="#text"/></item>
    <tag>out.mon=""</tag>
    <one-of>
        <item>january<tag>out.mon+="01";</tag></item>
        <item>february<tag>out.mon+="02";</tag></item>
        <item>march<tag>out.mon+="03";</tag></item>
        <item>april<tag>out.mon+="04";</tag></item>
        <item>may<tag>out.mon+="05";</tag></item>
        <item>june<tag>out.mon+="06";</tag></item>
        <item>july<tag>out.mon+="07";</tag></item>
        <item>august<tag>out.mon+="08";</tag></item>
        <item>september<tag>out.mon+="09";</tag></item>
        <item>october<tag>out.mon+="10";</tag></item>
        <item>november<tag>out.mon+="11";</tag></item>
        <item>december<tag>out.mon+="12";</tag></item>
        <item>jan<tag>out.mon+="01";</tag></item>
        <item>feb<tag>out.mon+="02";</tag></item>
        <item>aug<tag>out.mon+="08";</tag></item>
        <item>sept<tag>out.mon+="09";</tag></item>
        <item>oct<tag>out.mon+="10";</tag></item>
        <item>nov<tag>out.mon+="11";</tag></item>
        <item>dec<tag>out.mon+="12";</tag></item>
    </one-of>
  </rule>

  <rule id="digits">
      <item repeat="0-1"><ruleref uri="#text"/></item>
      <one-of>
          <item>0<tag>out=0;</tag></item>
          <item>1<tag>out=1;</tag></item>
          <item>2<tag>out=2;</tag></item>
          <item>3<tag>out=3;</tag></item>
          <item>4<tag>out=4;</tag></item>
          <item>5<tag>out=5;</tag></item>
          <item>6<tag>out=6;</tag></item>
          <item>7<tag>out=7;</tag></item>
          <item>8<tag>out=8;</tag></item>
          <item>9<tag>out=9;</tag></item>
          <item>first<tag>out=01;</tag></item>
          <item>second<tag>out=02;</tag></item>
          <item>third<tag>out=03;</tag></item>
```

```
            <item>fourth<tag>out=04;</tag></item>
            <item>fifth<tag>out=05;</tag></item>
            <item>sixth<tag>out=06;</tag></item>
            <item>seventh<tag>out=07;</tag></item>
            <item>eighth<tag>out=08;</tag></item>
            <item>ninth<tag>out=09;</tag></item>
            <item>one<tag>out=1;</tag></item>
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>

    <rule id="teens">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>ten<tag>out=10;</tag></item>
            <item>tenth<tag>out=10;</tag></item>
            <item>eleven<tag>out=11;</tag></item>
            <item>twelve<tag>out=12;</tag></item>
            <item>thirteen<tag>out=13;</tag></item>
            <item>fourteen<tag>out=14;</tag></item>
            <item>fifteen<tag>out=15;</tag></item>
            <item>sixteen<tag>out=16;</tag></item>
            <item>seventeen<tag>out=17;</tag></item>
            <item>eighteen<tag>out=18;</tag></item>
            <item>nineteen<tag>out=19;</tag></item>
            <item>tenth<tag>out=10;</tag></item>
            <item>eleventh<tag>out=11;</tag></item>
            <item>twelveth<tag>out=12;</tag></item>
            <item>thirteenth<tag>out=13;</tag></item>
            <item>fourteenth<tag>out=14;</tag></item>
            <item>fifteenth<tag>out=15;</tag></item>
            <item>sixteenth<tag>out=16;</tag></item>
            <item>seventeenth<tag>out=17;</tag></item>
            <item>eighteenth<tag>out=18;</tag></item>
            <item>nineteenth<tag>out=19;</tag></item>
        </one-of>
    </rule>
```

```
        <rule id="above_twenty">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
</grammar>
```

Montant de la réclamation

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: I want to make a claim of one hundre ten dollars
                 Output: $110

             Scenario 2:
                 Input: Requesting claim of Two hundred dollars
                 Output: $200

         -->

        <rule id="main" scope="public">
            <tag>out="$"</tag>
            <one-of>
                <item><ruleref uri="#sub_hundred"/><tag>out += rules.sub_hundred.sh;</
tag></item>
```

```
                <item><ruleref uri="#subThousands"/><tag>out += rules.subThousands;</
tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#thanks"/></item>
        </rule>

        <rule id="text">
           <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">I want to place a claim for</item>
              <item repeat="0-1">I want to make a claim of</item>
              <item repeat="0-1">I assess damage of</item>
              <item repeat="0-1">Requesting claim of</item>
           </one-of>
        </rule>

        <rule id="hesitation">
            <one-of>
               <item>Hmm</item>
               <item>Mmm</item>
               <item>My</item>
            </one-of>
          </rule>

          <rule id="thanks">
              <one-of>
                 <item>Thanks</item>
                 <item>I think</item>
              </one-of>
            </rule>

        <rule id="digits">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.num = 0;</tag>
            <one-of>
                <item>0<tag>out.num+=0;</tag></item>
                <item>1<tag>out.num+=1;</tag></item>
                <item>2<tag>out.num+=2;</tag></item>
                <item>3<tag>out.num+=3;</tag></item>
                <item>4<tag>out.num+=4;</tag></item>
                <item>5<tag>out.num+=5;</tag></item>
                <item>6<tag>out.num+=6;</tag></item>
                <item>7<tag>out.num+=7;</tag></item>
                <item>8<tag>out.num+=8;</tag></item>
```

```
            <item>9<tag>out.num+=9;</tag></item>
            <item>one<tag>out.num+=1;</tag></item>
            <item>two<tag>out.num+=2;</tag></item>
            <item>three<tag>out.num+=3;</tag></item>
            <item>four<tag>out.num+=4;</tag></item>
            <item>five<tag>out.num+=5;</tag></item>
            <item>six<tag>out.num+=6;</tag></item>
            <item>seven<tag>out.num+=7;</tag></item>
            <item>eight<tag>out.num+=8;</tag></item>
            <item>nine<tag>out.num+=9;</tag></item>
        </one-of>
        <item repeat="0-1"><ruleref uri="#currency"/></item>
    </rule>

    <rule id="teens">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <tag>out.teen = 0;</tag>
        <one-of>
            <item>ten<tag>out.teen+=10;</tag></item>
            <item>eleven<tag>out.teen+=11;</tag></item>
            <item>twelve<tag>out.teen+=12;</tag></item>
            <item>thirteen<tag>out.teen+=13;</tag></item>
            <item>fourteen<tag>out.teen+=14;</tag></item>
            <item>fifteen<tag>out.teen+=15;</tag></item>
            <item>sixteen<tag>out.teen+=16;</tag></item>
            <item>seventeen<tag>out.teen+=17;</tag></item>
            <item>eighteen<tag>out.teen+=18;</tag></item>
            <item>nineteen<tag>out.teen+=19;</tag></item>
        </one-of>
        <item repeat="0-1"><ruleref uri="#currency"/></item>
    </rule>

    <rule id="above_twenty">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <tag>out.tens = 0;</tag>
        <one-of>
            <item>twenty<tag>out.tens+=20;</tag></item>
            <item>thirty<tag>out.tens+=30;</tag></item>
            <item>forty<tag>out.tens+=40;</tag></item>
            <item>fifty<tag>out.tens+=50;</tag></item>
            <item>sixty<tag>out.tens+=60;</tag></item>
            <item>seventy<tag>out.tens+=70;</tag></item>
            <item>eighty<tag>out.tens+=80;</tag></item>
            <item>ninety<tag>out.tens+=90;</tag></item>
```

```
                <item>hundred<tag>out.tens+=100;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.tens +=
 rules.digits.num;</tag></item>
        </rule>

        <rule id="currency">
            <one-of>
                <item repeat="0-1">dollars</item>
                <item repeat="0-1">Dollars</item>
                <item repeat="0-1">dollar</item>
                <item repeat="0-1">Dollar</item>
            </one-of>
        </rule>


        <rule id="sub_hundred">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.sh = 0;</tag>
            <one-of>
                <item><ruleref uri="#teens"/><tag>out.sh += rules.teens.teen;</tag></
item>
                <item>
                    <ruleref uri="#above_twenty"/><tag>out.sh +=
 rules.above_twenty.tens;</tag>
                </item>
                <item><ruleref uri="#digits"/><tag>out.sh += rules.digits.num;</tag></
item>
            </one-of>
         </rule>

        <rule id="subThousands">
            <ruleref uri="#sub_hundred"/><tag>out = (100 * rules.sub_hundred.sh);</tag>
            hundred
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty.tens;</tag></item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out += rules.teens.teen;</
tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits.num;</
tag></item>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>
```

```
</grammar>
```

Montant de la prime

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Premium amounts
             Scenario 1:
                 Input: The premium for one hundre ten dollars
                 Output: $110

             Scenario 2:
                 Input: RPremium amount of Two hundred dollars
                 Output: $200

         -->

        <rule id="main" scope="public">
            <tag>out="$"</tag>
            <one-of>
                <item><ruleref uri="#sub_hundred"/><tag>out += rules.sub_hundred.sh;</
tag></item>
                <item><ruleref uri="#subThousands"/><tag>out += rules.subThousands;</
tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#thanks"/></item>
        </rule>

        <rule id="text">
           <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
```

```
            <item repeat="0-1">A premium of</item>
            <item repeat="0-1">Premium amount of</item>
            <item repeat="0-1">The premium for</item>
            <item repeat="0-1">Insurance premium for</item>
         </one-of>
      </rule>

      <rule id="hesitation">
          <one-of>
             <item>Hmm</item>
             <item>Mmm</item>
             <item>My</item>
          </one-of>
        </rule>

        <rule id="thanks">
            <one-of>
               <item>Thanks</item>
               <item>I think</item>
            </one-of>
          </rule>

      <rule id="digits">
          <item repeat="0-1"><ruleref uri="#text"/></item>
          <tag>out.num = 0;</tag>
          <one-of>
              <item>0<tag>out.num+=0;</tag></item>
              <item>1<tag>out.num+=1;</tag></item>
              <item>2<tag>out.num+=2;</tag></item>
              <item>3<tag>out.num+=3;</tag></item>
              <item>4<tag>out.num+=4;</tag></item>
              <item>5<tag>out.num+=5;</tag></item>
              <item>6<tag>out.num+=6;</tag></item>
              <item>7<tag>out.num+=7;</tag></item>
              <item>8<tag>out.num+=8;</tag></item>
              <item>9<tag>out.num+=9;</tag></item>
              <item>one<tag>out.num+=1;</tag></item>
              <item>two<tag>out.num+=2;</tag></item>
              <item>three<tag>out.num+=3;</tag></item>
              <item>four<tag>out.num+=4;</tag></item>
              <item>five<tag>out.num+=5;</tag></item>
              <item>six<tag>out.num+=6;</tag></item>
              <item>seven<tag>out.num+=7;</tag></item>
              <item>eight<tag>out.num+=8;</tag></item>
```

```
                <item>nine<tag>out.num+=9;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>

        <rule id="teens">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.teen = 0;</tag>
            <one-of>
                <item>ten<tag>out.teen+=10;</tag></item>
                <item>eleven<tag>out.teen+=11;</tag></item>
                <item>twelve<tag>out.teen+=12;</tag></item>
                <item>thirteen<tag>out.teen+=13;</tag></item>
                <item>fourteen<tag>out.teen+=14;</tag></item>
                <item>fifteen<tag>out.teen+=15;</tag></item>
                <item>sixteen<tag>out.teen+=16;</tag></item>
                <item>seventeen<tag>out.teen+=17;</tag></item>
                <item>eighteen<tag>out.teen+=18;</tag></item>
                <item>nineteen<tag>out.teen+=19;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>

        <rule id="above_twenty">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.tens = 0;</tag>
            <one-of>
                <item>twenty<tag>out.tens+=20;</tag></item>
                <item>thirty<tag>out.tens+=30;</tag></item>
                <item>forty<tag>out.tens+=40;</tag></item>
                <item>fifty<tag>out.tens+=50;</tag></item>
                <item>sixty<tag>out.tens+=60;</tag></item>
                <item>seventy<tag>out.tens+=70;</tag></item>
                <item>eighty<tag>out.tens+=80;</tag></item>
                <item>ninety<tag>out.tens+=90;</tag></item>
                <item>hundred<tag>out.tens+=100;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.tens +=
rules.digits.num;</tag></item>
        </rule>

        <rule id="currency">
            <one-of>
```

```
                <item repeat="0-1">dollars</item>
                <item repeat="0-1">Dollars</item>
                <item repeat="0-1">dollar</item>
                <item repeat="0-1">Dollar</item>
            </one-of>
        </rule>


        <rule id="sub_hundred">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.sh = 0;</tag>
            <one-of>
                <item><ruleref uri="#teens"/><tag>out.sh += rules.teens.teen;</tag></
item>
                <item>
                    <ruleref uri="#above_twenty"/><tag>out.sh +=
 rules.above_twenty.tens;</tag>
                </item>
                <item><ruleref uri="#digits"/><tag>out.sh += rules.digits.num;</tag></
item>
            </one-of>
         </rule>

        <rule id="subThousands">
            <ruleref uri="#sub_hundred"/><tag>out = (100 * rules.sub_hundred.sh);</tag>
            hundred
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty.tens;</tag></item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out += rules.teens.teen;</
tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits.num;</
tag></item>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>
</grammar>
```

Quantité de la politique

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
```

```
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: The number is one
                 Output: 1

             Scenario 2:
                 Input: I want policy for ten
                 Output: 10

         -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <one-of>
              <item repeat="1"><ruleref uri="#digits"/><tag>out+= rules.digits;</tag></
item>
              <item repeat="1"><ruleref uri="#teens"/><tag>out+= rules.teens;</tag></
item>
              <item repeat="1"><ruleref uri="#above_twenty"/><tag>out+=
 rules.above_twenty;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#thanks"/></item>
        </rule>

        <rule id="text">
           <one-of>
              <item repeat="0-1">I want policy for</item>
              <item repeat="0-1">I want to order policy for</item>
              <item repeat="0-1">The number is</item>
           </one-of>
        </rule>

        <rule id="thanks">
            <one-of>
               <item>Thanks</item>
               <item>I think</item>
```

```
        </one-of>
      </rule>

    <rule id="digits">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>0<tag>out=0;</tag></item>
            <item>1<tag>out=1;</tag></item>
            <item>2<tag>out=2;</tag></item>
            <item>3<tag>out=3;</tag></item>
            <item>4<tag>out=4;</tag></item>
            <item>5<tag>out=5;</tag></item>
            <item>6<tag>out=6;</tag></item>
            <item>7<tag>out=7;</tag></item>
            <item>8<tag>out=8;</tag></item>
            <item>9<tag>out=9;</tag></item>
            <item>one<tag>out=1;</tag></item>
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>

    <rule id="teens">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>ten<tag>out=10;</tag></item>
            <item>eleven<tag>out=11;</tag></item>
            <item>twelve<tag>out=12;</tag></item>
            <item>thirteen<tag>out=13;</tag></item>
            <item>fourteen<tag>out=14;</tag></item>
            <item>fifteen<tag>out=15;</tag></item>
            <item>sixteen<tag>out=16;</tag></item>
            <item>seventeen<tag>out=17;</tag></item>
            <item>eighteen<tag>out=18;</tag></item>
            <item>nineteen<tag>out=19;</tag></item>
            <item>10<tag>out=10;</tag></item>
            <item>11<tag>out=11;</tag></item>
            <item>12<tag>out=12;</tag></item>
```

```
                <item>13<tag>out=13;</tag></item>
                <item>14<tag>out=14;</tag></item>
                <item>15<tag>out=15;</tag></item>
                <item>16<tag>out=16;</tag></item>
                <item>17<tag>out=17;</tag></item>
                <item>18<tag>out=18;</tag></item>
                <item>19<tag>out=19;</tag></item>
            </one-of>
        </rule>

        <rule id="above_twenty">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
                <item>forty<tag>out=40;</tag></item>
                <item>fifty<tag>out=50;</tag></item>
                <item>sixty<tag>out=60;</tag></item>
                <item>seventy<tag>out=70;</tag></item>
                <item>eighty<tag>out=80;</tag></item>
                <item>ninety<tag>out=90;</tag></item>
                <item>20<tag>out=20;</tag></item>
                <item>30<tag>out=30;</tag></item>
                <item>40<tag>out=40;</tag></item>
                <item>50<tag>out=50;</tag></item>
                <item>60<tag>out=60;</tag></item>
                <item>70<tag>out=70;</tag></item>
                <item>80<tag>out=80;</tag></item>
                <item>90<tag>out=90;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>

</grammar>
```

Grammaires pour les télécoms (téléchargement)

Les grammaires suivantes sont prises en charge pour les télécommunications : numéro de téléphone, numéro de série, numéro de carte SIM, code postal américain, date d'expiration de la carte de crédit, dates de début, de renouvellement et d'expiration du plan, date de début du service, quantité d'équipement et montant de la facture.

Phone number (Numéro de téléphone)

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="digits"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support 10-12 digits number and here are couple of examples of
 valid inputs:

             Scenario 1:
                 Input: Mmm My phone number is two zero one two five two six seven
 eight five
                 Output: 2012526785

             Scenario 2:
                 Input: My phone number is two zero one two five two six seven eight
 five
                 Output: 2012526785

          -->

        <rule id="digits">
            <tag>out=""</tag>
            <item><ruleref uri="#singleDigit"/><tag>out += rules.singleDigit.digit;</
tag></item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">My phone number is</item>
              <item repeat="0-1">Phone number is</item>
              <item repeat="0-1">It is</item>
              <item repeat="0-1">Yes, it's</item>
              <item repeat="0-1">Yes, it is</item>
              <item repeat="0-1">Yes it is</item>
```

```
           </one-of>
        </rule>

        <rule id="hesitation">
            <one-of>
               <item>Hmm</item>
               <item>Mmm</item>
               <item>My</item>
            </one-of>
          </rule>

        <rule id="singleDigit">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.digit=""</tag>
            <item repeat="10-12">
                <one-of>
                    <item>0<tag>out.digit+=0;</tag></item>
                    <item>zero<tag>out.digit+=0;</tag></item>
                    <item>1<tag>out.digit+=1;</tag></item>
                    <item>one<tag>out.digit+=1;</tag></item>
                    <item>2<tag>out.digit+=2;</tag></item>
                    <item>two<tag>out.digit+=2;</tag></item>
                    <item>3<tag>out.digit+=3;</tag></item>
                    <item>three<tag>out.digit+=3;</tag></item>
                    <item>4<tag>out.digit+=4;</tag></item>
                    <item>four<tag>out.digit+=4;</tag></item>
                    <item>5<tag>out.digit+=5;</tag></item>
                    <item>five<tag>out.digit+=5;</tag></item>
                    <item>6<tag>out.digit+=6;</tag></item>
                    <item>six<tag>out.digit+=5;</tag></item>
                    <item>7<tag>out.digit+=7;</tag></item>
                    <item>seven<tag>out.digit+=7;</tag></item>
                    <item>8<tag>out.digit+=8;</tag></item>
                    <item>eight<tag>out.digit+=8;</tag></item>
                    <item>9<tag>out.digit+=9;</tag></item>
                    <item>nine<tag>out.digit+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

Numéro de série

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="digits"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: My serial number is 1 2 3 4 5 6 7 8 9 1 2 3 4 5 6
                 Output: 123456789123456

             Scenario 2:
                 Input: Device Serial number 1 2 3 4 5 6 7 8 9 1 2 3 4 5 6
                 Output: 123456789123456

         -->

        <rule id="digits">
            <tag>out=""</tag>
            <item><ruleref uri="#singleDigit"/><tag>out += rules.singleDigit.digit;</
tag></item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
            <one-of>
                <item repeat="0-1">My serial number is</item>
                <item repeat="0-1">Device Serial number</item>
                <item repeat="0-1">The number is</item>
                <item repeat="0-1">The IMEI number is</item>
            </one-of>
        </rule>

        <rule id="hesitation">
            <one-of>
```

```
            <item>Hmm</item>
            <item>Mmm</item>
            <item>My</item>
          </one-of>
        </rule>

      <rule id="singleDigit">
          <item repeat="0-1"><ruleref uri="#text"/></item>
          <tag>out.digit=""</tag>
          <item repeat="15">
              <one-of>
                  <item>0<tag>out.digit+=0;</tag></item>
                  <item>zero<tag>out.digit+=0;</tag></item>
                  <item>1<tag>out.digit+=1;</tag></item>
                  <item>one<tag>out.digit+=1;</tag></item>
                  <item>2<tag>out.digit+=2;</tag></item>
                  <item>two<tag>out.digit+=2;</tag></item>
                  <item>3<tag>out.digit+=3;</tag></item>
                  <item>three<tag>out.digit+=3;</tag></item>
                  <item>4<tag>out.digit+=4;</tag></item>
                  <item>four<tag>out.digit+=4;</tag></item>
                  <item>5<tag>out.digit+=5;</tag></item>
                  <item>five<tag>out.digit+=5;</tag></item>
                  <item>6<tag>out.digit+=6;</tag></item>
                  <item>six<tag>out.digit+=5;</tag></item>
                  <item>7<tag>out.digit+=7;</tag></item>
                  <item>seven<tag>out.digit+=7;</tag></item>
                  <item>8<tag>out.digit+=8;</tag></item>
                  <item>eight<tag>out.digit+=8;</tag></item>
                  <item>9<tag>out.digit+=9;</tag></item>
                  <item>nine<tag>out.digit+=9;</tag></item>
              </one-of>
          </item>
      </rule>
</grammar>
```

Numéro SIM

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
```

```
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">


        <!-- Test Cases

        Grammar will support the following inputs:

            Scenario 1:
                Input: My SIM number is A B C 1 2 3 4
                Output: ABC1234

            Scenario 2:
                Input: My SIM number is 1 2 3 4 A B C
                Output: 1234ABC

            Scenario 3:
                Input: My SIM number is 1 2 3 4 A B C 1
                Output: 123ABC1
        -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item><ruleref uri="#alphanumeric"/><tag>out +=
rules.alphanumeric.alphanum;</tag></item>
            <item repeat="0-1"><ruleref uri="#alphabets"/><tag>out +=
rules.alphabets.letters;</tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
rules.digits.numbers</tag></item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
            <one-of>
                <item repeat="0-1">My SIM number is</item>
                <item repeat="0-1">SIM number is</item>
            </one-of>
        </rule>

        <rule id="hesitation">
          <one-of>
             <item>Hmm</item>
```

```
            <item>Mmm</item>
            <item>My</item>
          </one-of>
        </rule>

        <rule id="alphanumeric" scope="public">
            <tag>out.alphanum=""</tag>
            <item><ruleref uri="#alphabets"/><tag>out.alphanum +=
rules.alphabets.letters;</tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.alphanum +=
rules.digits.numbers</tag></item>
        </rule>

        <rule id="alphabets">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.letters=""</tag>
            <tag>out.firstOccurence=""</tag>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.firstOccurence +=
rules.digits.numbers; out.letters += out.firstOccurence;</tag></item>
            <item repeat="1-">
                <one-of>
                    <item>A<tag>out.letters+='A';</tag></item>
                    <item>B<tag>out.letters+='B';</tag></item>
                    <item>C<tag>out.letters+='C';</tag></item>
                    <item>D<tag>out.letters+='D';</tag></item>
                    <item>E<tag>out.letters+='E';</tag></item>
                    <item>F<tag>out.letters+='F';</tag></item>
                    <item>G<tag>out.letters+='G';</tag></item>
                    <item>H<tag>out.letters+='H';</tag></item>
                    <item>I<tag>out.letters+='I';</tag></item>
                    <item>J<tag>out.letters+='J';</tag></item>
                    <item>K<tag>out.letters+='K';</tag></item>
                    <item>L<tag>out.letters+='L';</tag></item>
                    <item>M<tag>out.letters+='M';</tag></item>
                    <item>N<tag>out.letters+='N';</tag></item>
                    <item>O<tag>out.letters+='O';</tag></item>
                    <item>P<tag>out.letters+='P';</tag></item>
                    <item>Q<tag>out.letters+='Q';</tag></item>
                    <item>R<tag>out.letters+='R';</tag></item>
                    <item>S<tag>out.letters+='S';</tag></item>
                    <item>T<tag>out.letters+='T';</tag></item>
                    <item>U<tag>out.letters+='U';</tag></item>
                    <item>V<tag>out.letters+='V';</tag></item>
                    <item>W<tag>out.letters+='W';</tag></item>
```

```
                    <item>X<tag>out.letters+='X';</tag></item>
                    <item>Y<tag>out.letters+='Y';</tag></item>
                    <item>Z<tag>out.letters+='Z';</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="digits">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.numbers=""</tag>
            <item repeat="1-10">
                <one-of>
                    <item>0<tag>out.numbers+=0;</tag></item>
                    <item>1<tag>out.numbers+=1;</tag></item>
                    <item>2<tag>out.numbers+=2;</tag></item>
                    <item>3<tag>out.numbers+=3;</tag></item>
                    <item>4<tag>out.numbers+=4;</tag></item>
                    <item>5<tag>out.numbers+=5;</tag></item>
                    <item>6<tag>out.numbers+=6;</tag></item>
                    <item>7<tag>out.numbers+=7;</tag></item>
                    <item>8<tag>out.numbers+=8;</tag></item>
                    <item>9<tag>out.numbers+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

Code postal américain

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="digits"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support 5 digits code and here are couple of examples of valid
 inputs:
```

```
             Scenario 1:
                 Input: Mmmm My zipcode is umm One Oh Nine Eight Seven
                 Output: 10987

             Scenario 2:
                 Input: My zipcode is One Oh Nine Eight Seven
                 Output: 10987

         -->

        <rule id="digits">
            <tag>out=""</tag>
            <item><ruleref uri="#singleDigit"/><tag>out += rules.singleDigit.digit;</
tag></item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">My zipcode is</item>
              <item repeat="0-1">Zipcode is</item>
              <item repeat="0-1">It is</item>
           </one-of>
        </rule>

        <rule id="hesitation">
            <one-of>
               <item>Hmm</item>
               <item>Mmm</item>
               <item>My</item>
            </one-of>
          </rule>

        <rule id="singleDigit">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.digit=""</tag>
            <item repeat="5">
                <one-of>
                    <item>0<tag>out.digit+=0;</tag></item>
                    <item>zero<tag>out.digit+=0;</tag></item>
                    <item>Oh<tag>out.digit+=0;</tag></item>
                    <item>1<tag>out.digit+=1;</tag></item>
                    <item>one<tag>out.digit+=1;</tag></item>
```

```
                    <item>2<tag>out.digit+=2;</tag></item>
                    <item>two<tag>out.digit+=2;</tag></item>
                    <item>3<tag>out.digit+=3;</tag></item>
                    <item>three<tag>out.digit+=3;</tag></item>
                    <item>4<tag>out.digit+=4;</tag></item>
                    <item>four<tag>out.digit+=4;</tag></item>
                    <item>5<tag>out.digit+=5;</tag></item>
                    <item>five<tag>out.digit+=5;</tag></item>
                    <item>6<tag>out.digit+=6;</tag></item>
                    <item>six<tag>out.digit+=5;</tag></item>
                    <item>7<tag>out.digit+=7;</tag></item>
                    <item>seven<tag>out.digit+=7;</tag></item>
                    <item>8<tag>out.digit+=8;</tag></item>
                    <item>eight<tag>out.digit+=8;</tag></item>
                    <item>9<tag>out.digit+=9;</tag></item>
                    <item>nine<tag>out.digit+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

Date d'expiration de carte de crédit

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="dateCardExpiration"
         mode="voice"
         tag-format="semantics/1.0">

        <rule id="dateCardExpiration" scope="public">
            <tag>out="";</tag>
            <item repeat="1"><ruleref uri="#months"/><tag>out = out + rules.months;</
tag></item>
            <item repeat="1"><ruleref uri="#year"/><tag>out += " " + rules.year.yr;</
tag></item>
        </rule>

        <!-- Test Cases
```

```
        Grammar will support the following inputs:

            Scenario 1:
                Input: My card expiration date is july eleven
                Output: 07 2011

            Scenario 2:
                Input: My card expiration date is may twenty six
                Output: 05 2026

        -->

        <rule id="text">
           <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">My card expiration date is </item>
           </one-of>
        </rule>

        <rule id="hesitation">
          <one-of>
             <item>Hmm</item>
             <item>Mmm</item>
             <item>My</item>
          </one-of>
        </rule>

        <rule id="months">
           <item repeat="0-1"><ruleref uri="#text"/></item>
           <one-of>
               <item>january<tag>out="01";</tag></item>
               <item>february<tag>out="02";</tag></item>
               <item>march<tag>out="03";</tag></item>
               <item>april<tag>out="04";</tag></item>
               <item>may<tag>out="05";</tag></item>
               <item>june<tag>out="06";</tag></item>
               <item>july<tag>out="07";</tag></item>
               <item>august<tag>out="08";</tag></item>
               <item>september<tag>out="09";</tag></item>
               <item>october<tag>out="10";</tag></item>
               <item>november<tag>out="11";</tag></item>
               <item>december<tag>out="12";</tag></item>
               <item>jan<tag>out="01";</tag></item>
               <item>feb<tag>out="02";</tag></item>
```

```
            <item>aug<tag>out="08";</tag></item>
            <item>sept<tag>out="09";</tag></item>
            <item>oct<tag>out="10";</tag></item>
            <item>nov<tag>out="11";</tag></item>
            <item>dec<tag>out="12";</tag></item>
            <item>1<tag>out="01";</tag></item>
            <item>2<tag>out="02";</tag></item>
            <item>3<tag>out="03";</tag></item>
            <item>4<tag>out="04";</tag></item>
            <item>5<tag>out="05";</tag></item>
            <item>6<tag>out="06";</tag></item>
            <item>7<tag>out="07";</tag></item>
            <item>8<tag>out="08";</tag></item>
            <item>9<tag>out="09";</tag></item>
            <item>ten<tag>out="10";</tag></item>
            <item>eleven<tag>out="11";</tag></item>
            <item>twelve<tag>out="12";</tag></item>
        </one-of>
    </rule>

    <rule id="digits">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>0<tag>out=0;</tag></item>
            <item>1<tag>out=1;</tag></item>
            <item>2<tag>out=2;</tag></item>
            <item>3<tag>out=3;</tag></item>
            <item>4<tag>out=4;</tag></item>
            <item>5<tag>out=5;</tag></item>
            <item>6<tag>out=6;</tag></item>
            <item>7<tag>out=7;</tag></item>
            <item>8<tag>out=8;</tag></item>
            <item>9<tag>out=9;</tag></item>
            <item>one<tag>out=1;</tag></item>
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>
```

```
        <rule id="year">
          <tag>out.yr="20"</tag>
          <one-of>
              <item><ruleref uri="#teens"/><tag>out.yr += rules.teens;</tag></item>
              <item><ruleref uri="#above_twenty"/><tag>out.yr += rules.above_twenty;</
tag></item>
          </one-of>
        </rule>

        <rule id="teens">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>ten<tag>out=10;</tag></item>
                <item>eleven<tag>out=11;</tag></item>
                <item>twelve<tag>out=12;</tag></item>
                <item>thirteen<tag>out=13;</tag></item>
                <item>fourteen<tag>out=14;</tag></item>
                <item>fifteen<tag>out=15;</tag></item>
                <item>sixteen<tag>out=16;</tag></item>
                <item>seventeen<tag>out=17;</tag></item>
                <item>eighteen<tag>out=18;</tag></item>
                <item>nineteen<tag>out=19;</tag></item>
                <item>10<tag>out=10;</tag></item>
                <item>11<tag>out=11;</tag></item>
                <item>12<tag>out=12;</tag></item>
                <item>13<tag>out=13;</tag></item>
                <item>14<tag>out=14;</tag></item>
                <item>15<tag>out=15;</tag></item>
                <item>16<tag>out=16;</tag></item>
                <item>17<tag>out=17;</tag></item>
                <item>18<tag>out=18;</tag></item>
                <item>19<tag>out=19;</tag></item>
            </one-of>
        </rule>

        <rule id="above_twenty">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
                <item>forty<tag>out=40;</tag></item>
                <item>fifty<tag>out=50;</tag></item>
                <item>sixty<tag>out=60;</tag></item>
```

```
                <item>seventy<tag>out=70;</tag></item>
                <item>eighty<tag>out=80;</tag></item>
                <item>ninety<tag>out=90;</tag></item>
                <item>20<tag>out=20;</tag></item>
                <item>30<tag>out=30;</tag></item>
                <item>40<tag>out=40;</tag></item>
                <item>50<tag>out=50;</tag></item>
                <item>60<tag>out=60;</tag></item>
                <item>70<tag>out=70;</tag></item>
                <item>80<tag>out=80;</tag></item>
                <item>90<tag>out=90;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
</grammar>
```

Date d'expiration du plan, jour/mois/année

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: My plan expires on July Five Two Thousand and Eleven
                 Output: 07/5/11

             Scenario 2:
                 Input: My plan will expire on July Sixteen Two Thousand and Eleven
                 Output: 07/16/11

             Scenario 3:
                 Input: My plan will expire on July Thirty Two Thousand and Eleven
```

```
                Output: 07/30/11
        -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item>
                <item repeat="1"><ruleref uri="#months"/><tag>out = out +
 rules.months.mon + "/";</tag></item>
                <one-of>
                  <item><ruleref uri="#digits"/><tag>out += rules.digits + "/";</
tag></item>
                  <item><ruleref uri="#teens"/><tag>out += rules.teens+ "/";</tag></
item>
                  <item><ruleref uri="#above_twenty"/><tag>out += rules.above_twenty+
 "/";</tag></item>
                </one-of>
                <one-of>
                    <item><ruleref uri="#thousands"/><tag>out += rules.thousands;</
tag></item>
                    <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
 rules.digits;</tag></item>
                    <item repeat="0-1"><ruleref uri="#teens"/><tag>out +=
 rules.teens;</tag></item>
                    <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty;</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="text">
          <item repeat="0-1"><ruleref uri="#hesitation"/></item>
          <one-of>
               <item repeat="0-1">My plan expires on</item>
               <item repeat="0-1">My plan expired on</item>
               <item repeat="0-1">My plan will expire on</item>
           </one-of>
        </rule>

        <rule id="hesitation">
           <one-of>
              <item>Hmm</item>
              <item>Mmm</item>
              <item>My</item>
           </one-of>
```

```
 </rule>

<rule id="months">
   <tag>out.mon=""</tag>
<item repeat="0-1"><ruleref uri="#text"/></item>

   <one-of>
     <item>january<tag>out.mon+="01";</tag></item>
     <item>february<tag>out.mon+="02";</tag></item>
     <item>march<tag>out.mon+="03";</tag></item>
     <item>april<tag>out.mon+="04";</tag></item>
     <item>may<tag>out.mon+="05";</tag></item>
     <item>june<tag>out.mon+="06";</tag></item>
     <item>july<tag>out.mon+="07";</tag></item>
     <item>august<tag>out.mon+="08";</tag></item>
     <item>september<tag>out.mon+="09";</tag></item>
     <item>october<tag>out.mon+="10";</tag></item>
     <item>november<tag>out.mon+="11";</tag></item>
     <item>december<tag>out.mon+="12";</tag></item>
     <item>jan<tag>out.mon+="01";</tag></item>
     <item>feb<tag>out.mon+="02";</tag></item>
     <item>aug<tag>out.mon+="08";</tag></item>
     <item>sept<tag>out.mon+="09";</tag></item>
     <item>oct<tag>out.mon+="10";</tag></item>
     <item>nov<tag>out.mon+="11";</tag></item>
     <item>dec<tag>out.mon+="12";</tag></item>
   </one-of>
</rule>

<rule id="digits">
    <one-of>
        <item>zero<tag>out=0;</tag></item>
        <item>one<tag>out=1;</tag></item>
        <item>two<tag>out=2;</tag></item>
        <item>three<tag>out=3;</tag></item>
        <item>four<tag>out=4;</tag></item>
        <item>five<tag>out=5;</tag></item>
        <item>six<tag>out=6;</tag></item>
        <item>seven<tag>out=7;</tag></item>
        <item>eight<tag>out=8;</tag></item>
        <item>nine<tag>out=9;</tag></item>
    </one-of>
</rule>
```

```
        <rule id="teens">
            <one-of>
                <item>ten<tag>out=10;</tag></item>
                <item>eleven<tag>out=11;</tag></item>
                <item>twelve<tag>out=12;</tag></item>
                <item>thirteen<tag>out=13;</tag></item>
                <item>fourteen<tag>out=14;</tag></item>
                <item>fifteen<tag>out=15;</tag></item>
                <item>sixteen<tag>out=16;</tag></item>
                <item>seventeen<tag>out=17;</tag></item>
                <item>eighteen<tag>out=18;</tag></item>
                <item>nineteen<tag>out=19;</tag></item>
            </one-of>
        </rule>

        <rule id="thousands">
            <item>two thousand</item>
            <item repeat="0-1">and</item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out = rules.digits;</
tag></item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out = rules.teens;</tag></
item>
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out =
 rules.above_twenty;</tag></item>
        </rule>

        <rule id="above_twenty">
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
 </grammar>
```

Date de renouvellement du plan, mois/année

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                            http://www.w3.org/TR/speech-grammar/grammar.xsd"
```

```
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: My plan will renew on July Two Thousand and Eleven
                 Output: 07/11

             Scenario 2:
                 Input: Renew plan on July Two Thousand and Eleven
                 Output: 07/11

         -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item repeat="1-10">
                <item repeat="1"><ruleref uri="#months"/><tag>out = out +
 rules.months.mon + "/";</tag></item>
                <one-of>
                    <item><ruleref uri="#thousands"/><tag>out += rules.thousands;</
tag></item>
                    <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
 rules.digits;</tag></item>
                    <item repeat="0-1"><ruleref uri="#teens"/><tag>out +=
 rules.teens;</tag></item>
                    <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty;</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
            <one-of>
                <item repeat="0-1">My plan will renew on</item>
                <item repeat="0-1">My plan was renewed on</item>
                <item repeat="0-1">Renew plan on</item>
            </one-of>
```

```
        </rule>

        <rule id="hesitation">
            <one-of>
               <item>Hmm</item>
               <item>Mmm</item>
               <item>My</item>
            </one-of>
          </rule>

        <rule id="months">
           <item repeat="0-1"><ruleref uri="#text"/></item>
           <tag>out.mon=""</tag>
           <one-of>
               <item>january<tag>out.mon+="01";</tag></item>
               <item>february<tag>out.mon+="02";</tag></item>
               <item>march<tag>out.mon+="03";</tag></item>
               <item>april<tag>out.mon+="04";</tag></item>
               <item>may<tag>out.mon+="05";</tag></item>
               <item>june<tag>out.mon+="06";</tag></item>
               <item>july<tag>out.mon+="07";</tag></item>
               <item>august<tag>out.mon+="08";</tag></item>
               <item>september<tag>out.mon+="09";</tag></item>
               <item>october<tag>out.mon+="10";</tag></item>
               <item>november<tag>out.mon+="11";</tag></item>
               <item>december<tag>out.mon+="12";</tag></item>
               <item>jan<tag>out.mon+="01";</tag></item>
               <item>feb<tag>out.mon+="02";</tag></item>
               <item>aug<tag>out.mon+="08";</tag></item>
               <item>sept<tag>out.mon+="09";</tag></item>
               <item>oct<tag>out.mon+="10";</tag></item>
               <item>nov<tag>out.mon+="11";</tag></item>
               <item>dec<tag>out.mon+="12";</tag></item>
           </one-of>
        </rule>

        <rule id="digits">
            <one-of>
                <item>zero<tag>out=0;</tag></item>
                <item>one<tag>out=1;</tag></item>
                <item>two<tag>out=2;</tag></item>
                <item>three<tag>out=3;</tag></item>
                <item>four<tag>out=4;</tag></item>
                <item>five<tag>out=5;</tag></item>
```

```
                <item>six<tag>out=6;</tag></item>
                <item>seven<tag>out=7;</tag></item>
                <item>eight<tag>out=8;</tag></item>
                <item>nine<tag>out=9;</tag></item>
            </one-of>
        </rule>

        <rule id="teens">
            <one-of>
                <item>ten<tag>out=10;</tag></item>
                <item>eleven<tag>out=11;</tag></item>
                <item>twelve<tag>out=12;</tag></item>
                <item>thirteen<tag>out=13;</tag></item>
                <item>fourteen<tag>out=14;</tag></item>
                <item>fifteen<tag>out=15;</tag></item>
                <item>sixteen<tag>out=16;</tag></item>
                <item>seventeen<tag>out=17;</tag></item>
                <item>eighteen<tag>out=18;</tag></item>
                <item>nineteen<tag>out=19;</tag></item>
            </one-of>
        </rule>

        <rule id="thousands">
            <item>two thousand</item>
            <item repeat="0-1">and</item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out = rules.digits;</tag></
item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out = rules.teens;</tag></
item>
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out =
 rules.above_twenty;</tag></item>
        </rule>

        <rule id="above_twenty">
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
                <item>forty<tag>out=40;</tag></item>
                <item>fifty<tag>out=50;</tag></item>
                <item>sixty<tag>out=60;</tag></item>
                <item>seventy<tag>out=70;</tag></item>
                <item>eighty<tag>out=80;</tag></item>
                <item>ninety<tag>out=90;</tag></item>
            </one-of>
```

```
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
</grammar>
```

Date de début du plan, mois/jour

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: My plan will start on july twenty three
                 Output: 07/23

             Scenario 2:
                 Input: My plan will start on july fifteen
                 Output: 07/15

         -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item repeat="1-10">
                <item><ruleref uri="#months"/><tag>out= rules.months.mon + "/";</tag></
item>
                <one-of>
                    <item><ruleref uri="#digits"/><tag>out+= rules.digits;</tag></item>
                    <item><ruleref uri="#teens"/><tag>out+= rules.teens;</tag></item>
                    <item><ruleref uri="#above_twenty"/><tag>out+=
 rules.above_twenty;</tag></item>
                </one-of>
            </item>
```

```
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
             <one-of>
                <item repeat="0-1">My plan started on</item>
                <item repeat="0-1">My plan will start on</item>
                <item repeat="0-1">I paid it on</item>
                <item repeat="0-1">I paid bill for</item>
             </one-of>
          </rule>

          <rule id="hesitation">
             <one-of>
                <item>Hmm</item>
                <item>Mmm</item>
                <item>My</item>
             </one-of>
           </rule>

        <rule id="months">
           <item repeat="0-1"><ruleref uri="#text"/></item>
           <tag>out.mon=""</tag>
           <one-of>
               <item>january<tag>out.mon+="01";</tag></item>
               <item>february<tag>out.mon+="02";</tag></item>
               <item>march<tag>out.mon+="03";</tag></item>
               <item>april<tag>out.mon+="04";</tag></item>
               <item>may<tag>out.mon+="05";</tag></item>
               <item>june<tag>out.mon+="06";</tag></item>
               <item>july<tag>out.mon+="07";</tag></item>
               <item>august<tag>out.mon+="08";</tag></item>
               <item>september<tag>out.mon+="09";</tag></item>
               <item>october<tag>out.mon+="10";</tag></item>
               <item>november<tag>out.mon+="11";</tag></item>
               <item>december<tag>out.mon+="12";</tag></item>
               <item>jan<tag>out.mon+="01";</tag></item>
               <item>feb<tag>out.mon+="02";</tag></item>
               <item>aug<tag>out.mon+="08";</tag></item>
               <item>sept<tag>out.mon+="09";</tag></item>
               <item>oct<tag>out.mon+="10";</tag></item>
               <item>nov<tag>out.mon+="11";</tag></item>
               <item>dec<tag>out.mon+="12";</tag></item>
           </one-of>
```

```
    </rule>

    <rule id="digits">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>0<tag>out=0;</tag></item>
            <item>1<tag>out=1;</tag></item>
            <item>2<tag>out=2;</tag></item>
            <item>3<tag>out=3;</tag></item>
            <item>4<tag>out=4;</tag></item>
            <item>5<tag>out=5;</tag></item>
            <item>6<tag>out=6;</tag></item>
            <item>7<tag>out=7;</tag></item>
            <item>8<tag>out=8;</tag></item>
            <item>9<tag>out=9;</tag></item>
            <item>first<tag>out=01;</tag></item>
            <item>second<tag>out=02;</tag></item>
            <item>third<tag>out=03;</tag></item>
            <item>fourth<tag>out=04;</tag></item>
            <item>fifth<tag>out=05;</tag></item>
            <item>sixth<tag>out=06;</tag></item>
            <item>seventh<tag>out=07;</tag></item>
            <item>eighth<tag>out=08;</tag></item>
            <item>ninth<tag>out=09;</tag></item>
            <item>one<tag>out=1;</tag></item>
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>

    <rule id="teens">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>ten<tag>out=10;</tag></item>
            <item>tenth<tag>out=10;</tag></item>
            <item>eleven<tag>out=11;</tag></item>
            <item>twelve<tag>out=12;</tag></item>
            <item>thirteen<tag>out=13;</tag></item>
```

```
                <item>fourteen<tag>out=14;</tag></item>
                <item>fifteen<tag>out=15;</tag></item>
                <item>sixteen<tag>out=16;</tag></item>
                <item>seventeen<tag>out=17;</tag></item>
                <item>eighteen<tag>out=18;</tag></item>
                <item>nineteen<tag>out=19;</tag></item>
                <item>tenth<tag>out=10;</tag></item>
                <item>eleventh<tag>out=11;</tag></item>
                <item>twelveth<tag>out=12;</tag></item>
                <item>thirteenth<tag>out=13;</tag></item>
                <item>fourteenth<tag>out=14;</tag></item>
                <item>fifteenth<tag>out=15;</tag></item>
                <item>sixteenth<tag>out=16;</tag></item>
                <item>seventeenth<tag>out=17;</tag></item>
                <item>eighteenth<tag>out=18;</tag></item>
                <item>nineteenth<tag>out=19;</tag></item>
            </one-of>
        </rule>

        <rule id="above_twenty">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
</grammar>
```

Date de début du service, mois/jour

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases
```

```
         Grammar will support the following inputs:

             Scenario 1:
                 Input: My plan starts on july twenty three
                 Output: 07/23

             Scenario 2:
                 Input: I want to activate on july fifteen
                 Output: 07/15

         -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item repeat="1-10">
                <item><ruleref uri="#months"/><tag>out= rules.months.mon + "/";</tag></
item>
                <one-of>
                    <item><ruleref uri="#digits"/><tag>out+= rules.digits;</tag></item>
                    <item><ruleref uri="#teens"/><tag>out+= rules.teens;</tag></item>
                    <item><ruleref uri="#above_twenty"/><tag>out+=
 rules.above_twenty;</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
             <one-of>
                <item repeat="0-1">My plan starts on</item>
                <item repeat="0-1">I want to start my plan on</item>
                <item repeat="0-1">Activation date of</item>
                <item repeat="0-1">Start activation on</item>
                <item repeat="0-1">I want to activate on</item>
                <item repeat="0-1">Activate plan starting</item>
                <item repeat="0-1">Starting</item>
                <item repeat="0-1">Start on</item>
             </one-of>
          </rule>

          <rule id="hesitation">
             <one-of>
                <item>Hmm</item>
```

```
            <item>Mmm</item>
            <item>My</item>
          </one-of>
       </rule>

    <rule id="months">
       <item repeat="0-1"><ruleref uri="#text"/></item>
       <tag>out.mon=""</tag>
       <one-of>
           <item>january<tag>out.mon+="01";</tag></item>
           <item>february<tag>out.mon+="02";</tag></item>
           <item>march<tag>out.mon+="03";</tag></item>
           <item>april<tag>out.mon+="04";</tag></item>
           <item>may<tag>out.mon+="05";</tag></item>
           <item>june<tag>out.mon+="06";</tag></item>
           <item>july<tag>out.mon+="07";</tag></item>
           <item>august<tag>out.mon+="08";</tag></item>
           <item>september<tag>out.mon+="09";</tag></item>
           <item>october<tag>out.mon+="10";</tag></item>
           <item>november<tag>out.mon+="11";</tag></item>
           <item>december<tag>out.mon+="12";</tag></item>
           <item>jan<tag>out.mon+="01";</tag></item>
           <item>feb<tag>out.mon+="02";</tag></item>
           <item>aug<tag>out.mon+="08";</tag></item>
           <item>sept<tag>out.mon+="09";</tag></item>
           <item>oct<tag>out.mon+="10";</tag></item>
           <item>nov<tag>out.mon+="11";</tag></item>
           <item>dec<tag>out.mon+="12";</tag></item>
       </one-of>
    </rule>

    <rule id="digits">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>0<tag>out=0;</tag></item>
            <item>1<tag>out=1;</tag></item>
            <item>2<tag>out=2;</tag></item>
            <item>3<tag>out=3;</tag></item>
            <item>4<tag>out=4;</tag></item>
            <item>5<tag>out=5;</tag></item>
            <item>6<tag>out=6;</tag></item>
            <item>7<tag>out=7;</tag></item>
            <item>8<tag>out=8;</tag></item>
            <item>9<tag>out=9;</tag></item>
```

```
            <item>first<tag>out=01;</tag></item>
            <item>second<tag>out=02;</tag></item>
            <item>third<tag>out=03;</tag></item>
            <item>fourth<tag>out=04;</tag></item>
            <item>fifth<tag>out=05;</tag></item>
            <item>sixth<tag>out=06;</tag></item>
            <item>seventh<tag>out=07;</tag></item>
            <item>eighth<tag>out=08;</tag></item>
            <item>ninth<tag>out=09;</tag></item>
            <item>one<tag>out=1;</tag></item>
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>

    <rule id="teens">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>ten<tag>out=10;</tag></item>
            <item>tenth<tag>out=10;</tag></item>
            <item>eleven<tag>out=11;</tag></item>
            <item>twelve<tag>out=12;</tag></item>
            <item>thirteen<tag>out=13;</tag></item>
            <item>fourteen<tag>out=14;</tag></item>
            <item>fifteen<tag>out=15;</tag></item>
            <item>sixteen<tag>out=16;</tag></item>
            <item>seventeen<tag>out=17;</tag></item>
            <item>eighteen<tag>out=18;</tag></item>
            <item>nineteen<tag>out=19;</tag></item>
            <item>tenth<tag>out=10;</tag></item>
            <item>eleventh<tag>out=11;</tag></item>
            <item>twelveth<tag>out=12;</tag></item>
            <item>thirteenth<tag>out=13;</tag></item>
            <item>fourteenth<tag>out=14;</tag></item>
            <item>fifteenth<tag>out=15;</tag></item>
            <item>sixteenth<tag>out=16;</tag></item>
            <item>seventeenth<tag>out=17;</tag></item>
            <item>eighteenth<tag>out=18;</tag></item>
```

```
                <item>nineteenth<tag>out=19;</tag></item>
            </one-of>
        </rule>

        <rule id="above_twenty">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
</grammar>
```

Quantité d'équipement

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: The number is one
                 Output: 1

             Scenario 2:
                 Input: It is ten
                 Output: 10

         -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
```

```
            <one-of>
              <item repeat="1"><ruleref uri="#digits"/><tag>out+= rules.digits;</tag></
item>
              <item repeat="1"><ruleref uri="#teens"/><tag>out+= rules.teens;</tag></
item>
              <item repeat="1"><ruleref uri="#above_twenty"/><tag>out+=
 rules.above_twenty;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#thanks"/></item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">It is</item>
              <item repeat="0-1">The number is</item>
              <item repeat="0-1">Order</item>
              <item repeat="0-1">I want to order</item>
              <item repeat="0-1">Total equipment</item>
           </one-of>
        </rule>

        <rule id="hesitation">
            <one-of>
               <item>Hmm</item>
               <item>Mmm</item>
               <item>My</item>
            </one-of>
          </rule>

        <rule id="thanks">
            <one-of>
               <item>Thanks</item>
               <item>I think</item>
            </one-of>
          </rule>

        <rule id="digits">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>0<tag>out=0;</tag></item>
                <item>1<tag>out=1;</tag></item>
                <item>2<tag>out=2;</tag></item>
                <item>3<tag>out=3;</tag></item>
```

```
            <item>4<tag>out=4;</tag></item>
            <item>5<tag>out=5;</tag></item>
            <item>6<tag>out=6;</tag></item>
            <item>7<tag>out=7;</tag></item>
            <item>8<tag>out=8;</tag></item>
            <item>9<tag>out=9;</tag></item>
            <item>one<tag>out=1;</tag></item>
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>

    <rule id="teens">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>ten<tag>out=10;</tag></item>
            <item>eleven<tag>out=11;</tag></item>
            <item>twelve<tag>out=12;</tag></item>
            <item>thirteen<tag>out=13;</tag></item>
            <item>fourteen<tag>out=14;</tag></item>
            <item>fifteen<tag>out=15;</tag></item>
            <item>sixteen<tag>out=16;</tag></item>
            <item>seventeen<tag>out=17;</tag></item>
            <item>eighteen<tag>out=18;</tag></item>
            <item>nineteen<tag>out=19;</tag></item>
            <item>10<tag>out=10;</tag></item>
            <item>11<tag>out=11;</tag></item>
            <item>12<tag>out=12;</tag></item>
            <item>13<tag>out=13;</tag></item>
            <item>14<tag>out=14;</tag></item>
            <item>15<tag>out=15;</tag></item>
            <item>16<tag>out=16;</tag></item>
            <item>17<tag>out=17;</tag></item>
            <item>18<tag>out=18;</tag></item>
            <item>19<tag>out=19;</tag></item>
        </one-of>
    </rule>
```

```
        <rule id="above_twenty">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
                <item>forty<tag>out=40;</tag></item>
                <item>fifty<tag>out=50;</tag></item>
                <item>sixty<tag>out=60;</tag></item>
                <item>seventy<tag>out=70;</tag></item>
                <item>eighty<tag>out=80;</tag></item>
                <item>ninety<tag>out=90;</tag></item>
                <item>20<tag>out=20;</tag></item>
                <item>30<tag>out=30;</tag></item>
                <item>40<tag>out=40;</tag></item>
                <item>50<tag>out=50;</tag></item>
                <item>60<tag>out=60;</tag></item>
                <item>70<tag>out=70;</tag></item>
                <item>80<tag>out=80;</tag></item>
                <item>90<tag>out=90;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>

</grammar>
```

Montant de la facture

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

                 Input: I want to make a payment of one hundred ten dollars
```

```
                Output: $110

         -->

        <rule id="main" scope="public">
            <tag>out="$"</tag>
            <one-of>
                <item><ruleref uri="#sub_hundred"/><tag>out += rules.sub_hundred.sh;</
tag></item>
                <item><ruleref uri="#subThousands"/><tag>out += rules.subThousands;</
tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#thanks"/></item>
        </rule>

        <rule id="text">
           <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">I want to make a payment for</item>
              <item repeat="0-1">I want to make a payment of</item>
              <item repeat="0-1">Pay a total of</item>
              <item repeat="0-1">Paying</item>
              <item repeat="0-1">Pay bill for </item>
           </one-of>
        </rule>

        <rule id="hesitation">
            <one-of>
               <item>Hmm</item>
               <item>Mmm</item>
               <item>My</item>
            </one-of>
          </rule>

          <rule id="thanks">
              <one-of>
                 <item>Thanks</item>
                 <item>I think</item>
              </one-of>
            </rule>

        <rule id="digits">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.num = 0;</tag>
```

```
        <one-of>
            <item>0<tag>out.num+=0;</tag></item>
            <item>1<tag>out.num+=1;</tag></item>
            <item>2<tag>out.num+=2;</tag></item>
            <item>3<tag>out.num+=3;</tag></item>
            <item>4<tag>out.num+=4;</tag></item>
            <item>5<tag>out.num+=5;</tag></item>
            <item>6<tag>out.num+=6;</tag></item>
            <item>7<tag>out.num+=7;</tag></item>
            <item>8<tag>out.num+=8;</tag></item>
            <item>9<tag>out.num+=9;</tag></item>
            <item>one<tag>out.num+=1;</tag></item>
            <item>two<tag>out.num+=2;</tag></item>
            <item>three<tag>out.num+=3;</tag></item>
            <item>four<tag>out.num+=4;</tag></item>
            <item>five<tag>out.num+=5;</tag></item>
            <item>six<tag>out.num+=6;</tag></item>
            <item>seven<tag>out.num+=7;</tag></item>
            <item>eight<tag>out.num+=8;</tag></item>
            <item>nine<tag>out.num+=9;</tag></item>
        </one-of>
        <item repeat="0-1"><ruleref uri="#currency"/></item>
    </rule>

    <rule id="teens">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <tag>out.teen = 0;</tag>
        <one-of>
            <item>ten<tag>out.teen+=10;</tag></item>
            <item>eleven<tag>out.teen+=11;</tag></item>
            <item>twelve<tag>out.teen+=12;</tag></item>
            <item>thirteen<tag>out.teen+=13;</tag></item>
            <item>fourteen<tag>out.teen+=14;</tag></item>
            <item>fifteen<tag>out.teen+=15;</tag></item>
            <item>sixteen<tag>out.teen+=16;</tag></item>
            <item>seventeen<tag>out.teen+=17;</tag></item>
            <item>eighteen<tag>out.teen+=18;</tag></item>
            <item>nineteen<tag>out.teen+=19;</tag></item>
        </one-of>
        <item repeat="0-1"><ruleref uri="#currency"/></item>
    </rule>

    <rule id="above_twenty">
        <item repeat="0-1"><ruleref uri="#text"/></item>
```

```
            <tag>out.tens = 0;</tag>
            <one-of>
                <item>twenty<tag>out.tens+=20;</tag></item>
                <item>thirty<tag>out.tens+=30;</tag></item>
                <item>forty<tag>out.tens+=40;</tag></item>
                <item>fifty<tag>out.tens+=50;</tag></item>
                <item>sixty<tag>out.tens+=60;</tag></item>
                <item>seventy<tag>out.tens+=70;</tag></item>
                <item>eighty<tag>out.tens+=80;</tag></item>
                <item>ninety<tag>out.tens+=90;</tag></item>
                <item>hundred<tag>out.tens+=100;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.tens +=
 rules.digits.num;</tag></item>
        </rule>

        <rule id="currency">
            <one-of>
                <item repeat="0-1">dollars</item>
                <item repeat="0-1">Dollars</item>
                <item repeat="0-1">dollar</item>
                <item repeat="0-1">Dollar</item>
            </one-of>
        </rule>


        <rule id="sub_hundred">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.sh = 0;</tag>
            <one-of>
                <item><ruleref uri="#teens"/><tag>out.sh += rules.teens.teen;</tag></
item>
                <item>
                    <ruleref uri="#above_twenty"/><tag>out.sh +=
 rules.above_twenty.tens;</tag>
                </item>
                <item><ruleref uri="#digits"/><tag>out.sh += rules.digits.num;</tag></
item>
            </one-of>
         </rule>

        <rule id="subThousands">
            <ruleref uri="#sub_hundred"/><tag>out = (100 * rules.sub_hundred.sh);</tag>
```

```
            hundred
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty.tens;</tag></item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out += rules.teens.teen;</
tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits.num;</
tag></item>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>
</grammar>
```

Grammaires génériques (téléchargement)

Nous proposons les grammaires génériques suivantes : alphanumérique, devise, date (mm/jj/aa), chiffres, message d'accueil, hésitation et agent.

Alphanumérique

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">


        <!-- Test Cases

            Scenario 1:
                Input: A B C 1 2 3 4
                Output: ABC1234

            Scenario 2:
                Input: 1 2 3 4 A B C
                Output: 1234ABC

            Scenario 3:
                Input: 1 2 3 4 A B C 1
                Output: 123ABC1
        -->
```

```
        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item><ruleref uri="#alphanumeric"/><tag>out +=
rules.alphanumeric.alphanum;</tag></item>
            <item repeat="0-1"><ruleref uri="#alphabets"/><tag>out +=
rules.alphabets.letters;</tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
rules.digits.numbers</tag></item>
        </rule>

        <rule id="alphanumeric" scope="public">
            <tag>out.alphanum=""</tag>
            <item><ruleref uri="#alphabets"/><tag>out.alphanum +=
rules.alphabets.letters;</tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.alphanum +=
rules.digits.numbers</tag></item>
        </rule>

        <rule id="alphabets">
            <tag>out.letters=""</tag>
            <tag>out.firstOccurence=""</tag>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.firstOccurence +=
rules.digits.numbers; out.letters += out.firstOccurence;</tag></item>
            <item repeat="1-">
                <one-of>
                    <item>A<tag>out.letters+='A';</tag></item>
                    <item>B<tag>out.letters+='B';</tag></item>
                    <item>C<tag>out.letters+='C';</tag></item>
                    <item>D<tag>out.letters+='D';</tag></item>
                    <item>E<tag>out.letters+='E';</tag></item>
                    <item>F<tag>out.letters+='F';</tag></item>
                    <item>G<tag>out.letters+='G';</tag></item>
                    <item>H<tag>out.letters+='H';</tag></item>
                    <item>I<tag>out.letters+='I';</tag></item>
                    <item>J<tag>out.letters+='J';</tag></item>
                    <item>K<tag>out.letters+='K';</tag></item>
                    <item>L<tag>out.letters+='L';</tag></item>
                    <item>M<tag>out.letters+='M';</tag></item>
                    <item>N<tag>out.letters+='N';</tag></item>
                    <item>O<tag>out.letters+='O';</tag></item>
                    <item>P<tag>out.letters+='P';</tag></item>
                    <item>Q<tag>out.letters+='Q';</tag></item>
                    <item>R<tag>out.letters+='R';</tag></item>
                    <item>S<tag>out.letters+='S';</tag></item>
```

```
                    <item>T<tag>out.letters+='T';</tag></item>
                    <item>U<tag>out.letters+='U';</tag></item>
                    <item>V<tag>out.letters+='V';</tag></item>
                    <item>W<tag>out.letters+='W';</tag></item>
                    <item>X<tag>out.letters+='X';</tag></item>
                    <item>Y<tag>out.letters+='Y';</tag></item>
                    <item>Z<tag>out.letters+='Z';</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="digits">
            <tag>out.numbers="";</tag>
            <item repeat="1-10">
                <one-of>
                    <item>0<tag>out.numbers+=0;</tag></item>
                    <item>1<tag>out.numbers+=1;</tag></item>
                    <item>2<tag>out.numbers+=2;</tag></item>
                    <item>3<tag>out.numbers+=3;</tag></item>
                    <item>4<tag>out.numbers+=4;</tag></item>
                    <item>5<tag>out.numbers+=5;</tag></item>
                    <item>6<tag>out.numbers+=6;</tag></item>
                    <item>7<tag>out.numbers+=7;</tag></item>
                    <item>8<tag>out.numbers+=8;</tag></item>
                    <item>9<tag>out.numbers+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

Devise

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

        <rule id="main" scope="public">
```

```
            <tag>out="$"</tag>
            <one-of>
                <item><ruleref uri="#sub_hundred"/><tag>out += rules.sub_hundred.sh;</
tag></item>
                <item><ruleref uri="#subThousands"/><tag>out += rules.subThousands;</
tag></item>
            </one-of>
        </rule>

        <rule id="digits">
            <tag>out.num = 0;</tag>
            <one-of>
                <item>0<tag>out.num+=0;</tag></item>
                <item>1<tag>out.num+=1;</tag></item>
                <item>2<tag>out.num+=2;</tag></item>
                <item>3<tag>out.num+=3;</tag></item>
                <item>4<tag>out.num+=4;</tag></item>
                <item>5<tag>out.num+=5;</tag></item>
                <item>6<tag>out.num+=6;</tag></item>
                <item>7<tag>out.num+=7;</tag></item>
                <item>8<tag>out.num+=8;</tag></item>
                <item>9<tag>out.num+=9;</tag></item>
                <item>one<tag>out.num+=1;</tag></item>
                <item>two<tag>out.num+=2;</tag></item>
                <item>three<tag>out.num+=3;</tag></item>
                <item>four<tag>out.num+=4;</tag></item>
                <item>five<tag>out.num+=5;</tag></item>
                <item>six<tag>out.num+=6;</tag></item>
                <item>seven<tag>out.num+=7;</tag></item>
                <item>eight<tag>out.num+=8;</tag></item>
                <item>nine<tag>out.num+=9;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>

        <rule id="teens">
            <tag>out.teen = 0;</tag>
            <one-of>
                <item>ten<tag>out.teen+=10;</tag></item>
                <item>eleven<tag>out.teen+=11;</tag></item>
                <item>twelve<tag>out.teen+=12;</tag></item>
                <item>thirteen<tag>out.teen+=13;</tag></item>
                <item>fourteen<tag>out.teen+=14;</tag></item>
                <item>fifteen<tag>out.teen+=15;</tag></item>
```

```
                <item>sixteen<tag>out.teen+=16;</tag></item>
                <item>seventeen<tag>out.teen+=17;</tag></item>
                <item>eighteen<tag>out.teen+=18;</tag></item>
                <item>nineteen<tag>out.teen+=19;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>

        <rule id="above_twenty">
            <tag>out.tens = 0;</tag>
            <one-of>
                <item>twenty<tag>out.tens+=20;</tag></item>
                <item>thirty<tag>out.tens+=30;</tag></item>
                <item>forty<tag>out.tens+=40;</tag></item>
                <item>fifty<tag>out.tens+=50;</tag></item>
                <item>sixty<tag>out.tens+=60;</tag></item>
                <item>seventy<tag>out.tens+=70;</tag></item>
                <item>eighty<tag>out.tens+=80;</tag></item>
                <item>ninety<tag>out.tens+=90;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.tens +=
 rules.digits.num;</tag></item>
        </rule>

        <rule id="currency">
            <one-of>
                <item repeat="0-1">dollars</item>
                <item repeat="0-1">Dollars</item>
                <item repeat="0-1">dollar</item>
                <item repeat="0-1">Dollar</item>
            </one-of>
        </rule>


        <rule id="sub_hundred">
            <tag>out.sh = 0;</tag>
            <one-of>
                <item><ruleref uri="#teens"/><tag>out.sh += rules.teens.teen;</tag></
item>
                <item>
                    <ruleref uri="#above_twenty"/><tag>out.sh +=
 rules.above_twenty.tens;</tag>
                </item>
```

```
                <item><ruleref uri="#digits"/><tag>out.sh += rules.digits.num;</tag></
item>
            </one-of>
         </rule>

        <rule id="subThousands">
            <ruleref uri="#sub_hundred"/><tag>out = (100 * rules.sub_hundred.sh);</tag>
            hundred
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty.tens;</tag></item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out += rules.teens.teen;</
tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits.num;</
tag></item>
        </rule>
</grammar>
```

Date, jj/mm

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item repeat="1-10">
                <one-of>
                    <item><ruleref uri="#digits"/><tag>out += rules.digits + " ";</
tag></item>
                    <item><ruleref uri="#teens"/><tag>out += rules.teens+ " ";</tag></
item>
                    <item><ruleref uri="#above_twenty"/><tag>out += rules.above_twenty+
 " ";</tag></item>
                </one-of>
                <item><ruleref uri="#months"/><tag>out = out + rules.months;</tag></
item>
            </item>
```

```
        </rule>

        <rule id="months">
           <one-of>
             <item>january<tag>out="january";</tag></item>
             <item>february<tag>out="february";</tag></item>
             <item>march<tag>out="march";</tag></item>
             <item>april<tag>out="april";</tag></item>
             <item>may<tag>out="may";</tag></item>
             <item>june<tag>out="june";</tag></item>
             <item>july<tag>out="july";</tag></item>
             <item>august<tag>out="august";</tag></item>
             <item>september<tag>out="september";</tag></item>
             <item>october<tag>out="october";</tag></item>
             <item>november<tag>out="november";</tag></item>
             <item>december<tag>out="december";</tag></item>
             <item>jan<tag>out="january";</tag></item>
             <item>feb<tag>out="february";</tag></item>
             <item>aug<tag>out="august";</tag></item>
             <item>sept<tag>out="september";</tag></item>
             <item>oct<tag>out="october";</tag></item>
             <item>nov<tag>out="november";</tag></item>
             <item>dec<tag>out="december";</tag></item>
           </one-of>
       </rule>

        <rule id="digits">
            <one-of>
                <item>0<tag>out=0;</tag></item>
                <item>1<tag>out=1;</tag></item>
                <item>2<tag>out=2;</tag></item>
                <item>3<tag>out=3;</tag></item>
                <item>4<tag>out=4;</tag></item>
                <item>5<tag>out=5;</tag></item>
                <item>6<tag>out=6;</tag></item>
                <item>7<tag>out=7;</tag></item>
                <item>8<tag>out=8;</tag></item>
                <item>9<tag>out=9;</tag></item>
                <item>first<tag>out=1;</tag></item>
                <item>second<tag>out=2;</tag></item>
                <item>third<tag>out=3;</tag></item>
                <item>fourth<tag>out=4;</tag></item>
                <item>fifth<tag>out=5;</tag></item>
                <item>sixth<tag>out=6;</tag></item>
```

```
            <item>seventh<tag>out=7;</tag></item>
            <item>eighth<tag>out=8;</tag></item>
            <item>ninth<tag>out=9;</tag></item>
            <item>one<tag>out=1;</tag></item>
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>

    <rule id="teens">
        <one-of>
            <item>ten<tag>out=10;</tag></item>
            <item>tenth<tag>out=10;</tag></item>
            <item>eleven<tag>out=11;</tag></item>
            <item>twelve<tag>out=12;</tag></item>
            <item>thirteen<tag>out=13;</tag></item>
            <item>fourteen<tag>out=14;</tag></item>
            <item>fifteen<tag>out=15;</tag></item>
            <item>sixteen<tag>out=16;</tag></item>
            <item>seventeen<tag>out=17;</tag></item>
            <item>eighteen<tag>out=18;</tag></item>
            <item>nineteen<tag>out=19;</tag></item>
            <item>tenth<tag>out=10;</tag></item>
            <item>eleventh<tag>out=11;</tag></item>
            <item>twelveth<tag>out=12;</tag></item>
            <item>thirteenth<tag>out=13;</tag></item>
            <item>fourteenth<tag>out=14;</tag></item>
            <item>fifteenth<tag>out=15;</tag></item>
            <item>sixteenth<tag>out=16;</tag></item>
            <item>seventeenth<tag>out=17;</tag></item>
            <item>eighteenth<tag>out=18;</tag></item>
            <item>nineteenth<tag>out=19;</tag></item>
        </one-of>
    </rule>

    <rule id="above_twenty">
        <one-of>
            <item>twenty<tag>out=20;</tag></item>
```

```
                <item>thirty<tag>out=30;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
</grammar>
```

Date, mm/aa

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item repeat="1-10">
                <item repeat="1"><ruleref uri="#months"/><tag>out = out +
 rules.months.mon + " ";</tag></item>
                <one-of>
                    <item><ruleref uri="#thousands"/><tag>out += rules.thousands;</
tag></item>
                    <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
 rules.digits;</tag></item>
                    <item repeat="0-1"><ruleref uri="#teens"/><tag>out +=
 rules.teens;</tag></item>
                    <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty;</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="months">
           <tag>out.mon=""</tag>
           <one-of>
               <item>january<tag>out.mon+="january";</tag></item>
               <item>february<tag>out.mon+="february";</tag></item>
               <item>march<tag>out.mon+="march";</tag></item>
```

```
            <item>april<tag>out.mon+="april";</tag></item>
            <item>may<tag>out.mon+="may";</tag></item>
            <item>june<tag>out.mon+="june";</tag></item>
            <item>july<tag>out.mon+="july";</tag></item>
            <item>august<tag>out.mon+="august";</tag></item>
            <item>september<tag>out.mon+="september";</tag></item>
            <item>october<tag>out.mon+="october";</tag></item>
            <item>november<tag>out.mon+="november";</tag></item>
            <item>december<tag>out.mon+="december";</tag></item>
            <item>jan<tag>out.mon+="january";</tag></item>
            <item>feb<tag>out.mon+="february";</tag></item>
            <item>aug<tag>out.mon+="august";</tag></item>
            <item>sept<tag>out.mon+="september";</tag></item>
            <item>oct<tag>out.mon+="october";</tag></item>
            <item>nov<tag>out.mon+="november";</tag></item>
            <item>dec<tag>out.mon+="december";</tag></item>
        </one-of>
    </rule>

    <rule id="digits">
        <one-of>
            <item>zero<tag>out=0;</tag></item>
            <item>one<tag>out=1;</tag></item>
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>

    <rule id="teens">
        <one-of>
            <item>ten<tag>out=10;</tag></item>
            <item>eleven<tag>out=11;</tag></item>
            <item>twelve<tag>out=12;</tag></item>
            <item>thirteen<tag>out=13;</tag></item>
            <item>fourteen<tag>out=14;</tag></item>
            <item>fifteen<tag>out=15;</tag></item>
            <item>sixteen<tag>out=16;</tag></item>
            <item>seventeen<tag>out=17;</tag></item>
```

```
                <item>eighteen<tag>out=18;</tag></item>
                <item>nineteen<tag>out=19;</tag></item>
            </one-of>
        </rule>

       <!--  <rule id="singleDigit">
            <item><ruleref uri="#digits"/><tag>out += rules.digits;</tag></item>
        </rule> -->

        <rule id="thousands">
            <!-- <item>
                <ruleref uri="#digits"/>
                <tag>out = (1000 * rules.digits);</tag>
                thousand
            </item> -->
            <item>two thousand<tag>out=2000;</tag></item>
            <item repeat="0-1">and</item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out += rules.teens;</tag></
item>
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty;</tag></item>
        </rule>

        <rule id="above_twenty">
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
                <item>forty<tag>out=40;</tag></item>
                <item>fifty<tag>out=50;</tag></item>
                <item>sixty<tag>out=60;</tag></item>
                <item>seventy<tag>out=70;</tag></item>
                <item>eighty<tag>out=80;</tag></item>
                <item>ninety<tag>out=90;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>

</grammar>
```

Date, jj/mm/aaaa

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <rule id="main" scope="public">
             <tag>out=""</tag>
             <item repeat="1-10">
                 <one-of>
                   <item><ruleref uri="#digits"/><tag>out += rules.digits + " ";</
tag></item>
                   <item><ruleref uri="#teens"/><tag>out += rules.teens+ " ";</tag></
item>
                   <item><ruleref uri="#above_twenty"/><tag>out += rules.above_twenty+
 " ";</tag></item>
                 </one-of>
                 <item repeat="1"><ruleref uri="#months"/><tag>out = out +
 rules.months.mon + " ";</tag></item>
                 <one-of>
                     <item><ruleref uri="#thousands"/><tag>out += rules.thousands;</
tag></item>
                     <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
 rules.digits;</tag></item>
                     <item repeat="0-1"><ruleref uri="#teens"/><tag>out +=
 rules.teens;</tag></item>
                     <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty;</tag></item>
                 </one-of>
             </item>
         </rule>

         <rule id="months">
            <tag>out.mon=""</tag>
            <one-of>
                <item>january<tag>out.mon+="january";</tag></item>
                <item>february<tag>out.mon+="february";</tag></item>
                <item>march<tag>out.mon+="march";</tag></item>
```

```
            <item>april<tag>out.mon+="april";</tag></item>
            <item>may<tag>out.mon+="may";</tag></item>
            <item>june<tag>out.mon+="june";</tag></item>
            <item>july<tag>out.mon+="july";</tag></item>
            <item>august<tag>out.mon+="august";</tag></item>
            <item>september<tag>out.mon+="september";</tag></item>
            <item>october<tag>out.mon+="october";</tag></item>
            <item>november<tag>out.mon+="november";</tag></item>
            <item>december<tag>out.mon+="december";</tag></item>
            <item>jan<tag>out.mon+="january";</tag></item>
            <item>feb<tag>out.mon+="february";</tag></item>
            <item>aug<tag>out.mon+="august";</tag></item>
            <item>sept<tag>out.mon+="september";</tag></item>
            <item>oct<tag>out.mon+="october";</tag></item>
            <item>nov<tag>out.mon+="november";</tag></item>
            <item>dec<tag>out.mon+="december";</tag></item>
        </one-of>
    </rule>

    <rule id="digits">
        <one-of>
            <item>zero<tag>out=0;</tag></item>
            <item>one<tag>out=1;</tag></item>
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>

    <rule id="teens">
        <one-of>
            <item>ten<tag>out=10;</tag></item>
            <item>eleven<tag>out=11;</tag></item>
            <item>twelve<tag>out=12;</tag></item>
            <item>thirteen<tag>out=13;</tag></item>
            <item>fourteen<tag>out=14;</tag></item>
            <item>fifteen<tag>out=15;</tag></item>
            <item>sixteen<tag>out=16;</tag></item>
            <item>seventeen<tag>out=17;</tag></item>
```

```
                <item>eighteen<tag>out=18;</tag></item>
                <item>nineteen<tag>out=19;</tag></item>
            </one-of>
        </rule>

        <rule id="thousands">
            <item>two thousand<tag>out=2000;</tag></item>
            <item repeat="0-1">and</item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out += rules.teens;</tag></
item>
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty;</tag></item>
        </rule>

        <rule id="above_twenty">
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
                <item>forty<tag>out=40;</tag></item>
                <item>fifty<tag>out=50;</tag></item>
                <item>sixty<tag>out=60;</tag></item>
                <item>seventy<tag>out=70;</tag></item>
                <item>eighty<tag>out=80;</tag></item>
                <item>ninety<tag>out=90;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>

 </grammar>
```

Chiffres, chiffres

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                            http://www.w3.org/TR/speech-grammar/grammar.xsd"
        xml:lang="en-US" version="1.0"
        root="digits"
        mode="voice"
```

```
         tag-format="semantics/1.0">

        <rule id="digits">
            <tag>out=""</tag>
            <item><ruleref uri="#singleDigit"/><tag>out += rules.singleDigit.digit;</
tag></item>
        </rule>

        <rule id="singleDigit">
            <tag>out.digit=""</tag>
            <item repeat="1-10">
                <one-of>
                    <item>0<tag>out.digit+=0;</tag></item>
                    <item>zero<tag>out.digit+=0;</tag></item>
                    <item>1<tag>out.digit+=1;</tag></item>
                    <item>one<tag>out.digit+=1;</tag></item>
                    <item>2<tag>out.digit+=2;</tag></item>
                    <item>two<tag>out.digit+=2;</tag></item>
                    <item>3<tag>out.digit+=3;</tag></item>
                    <item>three<tag>out.digit+=3;</tag></item>
                    <item>4<tag>out.digit+=4;</tag></item>
                    <item>four<tag>out.digit+=4;</tag></item>
                    <item>5<tag>out.digit+=5;</tag></item>
                    <item>five<tag>out.digit+=5;</tag></item>
                    <item>6<tag>out.digit+=6;</tag></item>
                    <item>six<tag>out.digit+=6;</tag></item>
                    <item>7<tag>out.digit+=7;</tag></item>
                    <item>seven<tag>out.digit+=7;</tag></item>
                    <item>8<tag>out.digit+=8;</tag></item>
                    <item>eight<tag>out.digit+=8;</tag></item>
                    <item>9<tag>out.digit+=9;</tag></item>
                    <item>nine<tag>out.digit+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

Nombres ordinaux

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
```

```
                              http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <one-of>
              <item repeat="1"><ruleref uri="#digits"/><tag>out+= rules.digits;</tag></
item>
              <item repeat="1"><ruleref uri="#teens"/><tag>out+= rules.teens;</tag></
item>
              <item repeat="1"><ruleref uri="#above_twenty"/><tag>out+=
 rules.above_twenty;</tag></item>
            </one-of>
        </rule>

        <rule id="digits">
            <one-of>
                <item>0<tag>out=0;</tag></item>
                <item>1<tag>out=1;</tag></item>
                <item>2<tag>out=2;</tag></item>
                <item>3<tag>out=3;</tag></item>
                <item>4<tag>out=4;</tag></item>
                <item>5<tag>out=5;</tag></item>
                <item>6<tag>out=6;</tag></item>
                <item>7<tag>out=7;</tag></item>
                <item>8<tag>out=8;</tag></item>
                <item>9<tag>out=9;</tag></item>
                <item>one<tag>out=1;</tag></item>
                <item>two<tag>out=2;</tag></item>
                <item>three<tag>out=3;</tag></item>
                <item>four<tag>out=4;</tag></item>
                <item>five<tag>out=5;</tag></item>
                <item>six<tag>out=6;</tag></item>
                <item>seven<tag>out=7;</tag></item>
                <item>eight<tag>out=8;</tag></item>
                <item>nine<tag>out=9;</tag></item>
            </one-of>
        </rule>

        <rule id="teens">
            <one-of>
```

```
                <item>ten<tag>out=10;</tag></item>
                <item>eleven<tag>out=11;</tag></item>
                <item>twelve<tag>out=12;</tag></item>
                <item>thirteen<tag>out=13;</tag></item>
                <item>fourteen<tag>out=14;</tag></item>
                <item>fifteen<tag>out=15;</tag></item>
                <item>sixteen<tag>out=16;</tag></item>
                <item>seventeen<tag>out=17;</tag></item>
                <item>eighteen<tag>out=18;</tag></item>
                <item>nineteen<tag>out=19;</tag></item>
                <item>10<tag>out=10;</tag></item>
                <item>11<tag>out=11;</tag></item>
                <item>12<tag>out=12;</tag></item>
                <item>13<tag>out=13;</tag></item>
                <item>14<tag>out=14;</tag></item>
                <item>15<tag>out=15;</tag></item>
                <item>16<tag>out=16;</tag></item>
                <item>17<tag>out=17;</tag></item>
                <item>18<tag>out=18;</tag></item>
                <item>19<tag>out=19;</tag></item>
            </one-of>
        </rule>

        <rule id="above_twenty">
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
                <item>forty<tag>out=40;</tag></item>
                <item>fifty<tag>out=50;</tag></item>
                <item>sixty<tag>out=60;</tag></item>
                <item>seventy<tag>out=70;</tag></item>
                <item>eighty<tag>out=80;</tag></item>
                <item>ninety<tag>out=90;</tag></item>
                <item>20<tag>out=20;</tag></item>
                <item>30<tag>out=30;</tag></item>
                <item>40<tag>out=40;</tag></item>
                <item>50<tag>out=50;</tag></item>
                <item>60<tag>out=60;</tag></item>
                <item>70<tag>out=70;</tag></item>
                <item>80<tag>out=80;</tag></item>
                <item>90<tag>out=90;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
```

```
        </rule>

</grammar>
```

Agent

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <rule id="main" scope="public">
             <tag>out=""</tag>
             <ruleref uri="#text"/><tag>out = rules.text</tag>
         </rule>

         <rule id="text">
           <one-of>
              <item>Can I talk to the agent<tag>out="You will be trasnfered to the
 agent in a while"</tag></item>
              <item>talk to an agent<tag>out="You will be trasnfered to the agent in a
 while"</tag></item>
           </one-of>
         </rule>
</grammar>
```

Salut

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">
```

```
         <rule id="main" scope="public">
             <tag>out=""</tag>
             <ruleref uri="#text"/><tag>out = rules.text</tag>
         </rule>

         <rule id="text">
           <one-of>
              <item>hey<tag>out="Greeting"</tag></item>
              <item>hi<tag>out="Greeting"</tag></item>
              <item>Hi<tag>out="Greeting"</tag></item>
              <item>Hey<tag>out="Greeting"</tag></item>
              <item>Hello<tag>out="Greeting"</tag></item>
              <item>hello<tag>out="Greeting"</tag></item>
           </one-of>
         </rule>
</grammar>
```

Hésitation

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <rule id="main" scope="public">
             <tag>out=""</tag>
             <ruleref uri="#text"/><tag>out = rules.text</tag>
         </rule>

         <rule id="text">
           <one-of>
              <item>Hmm<tag>out="Waiting for your input"</tag></item>
              <item>Mmm<tag>out="Waiting for your input"</tag></item>
              <item>Can you please wait<tag>out="Waiting for your input"</tag></item>
           </one-of>
         </rule>
</grammar>
```

Type de fente composite

Un emplacement composite est une combinaison de deux emplacements ou plus qui capturent plusieurs informations dans une seule entrée utilisateur. Par exemple, vous pouvez configurer le bot pour obtenir la position en demandant « la ville et l'État ou le code postal ». En revanche, lorsque la conversation est configurée pour utiliser des types de créneaux distincts, ce qui entraîne une expérience de conversation rigide (« Qu'est-ce que la ville ? » suivi de « Qu'est-ce que le code postal ? »). Avec un emplacement composite, vous pouvez capturer toutes les informations via un seul emplacement. Un emplacement composite est une combinaison d'emplacements appelés sous-emplacements, tels que la ville, l'État et le code postal.

Vous pouvez utiliser une combinaison des types d'emplacements Amazon Lex disponibles (intégrés) et vos propres emplacements (emplacements personnalisés). Vous pouvez concevoir des expressions logiques pour capturer des informations dans les sous-emplacements requis. Par exemple : ville et état ou code postal.

Le type d'emplacement composite n'est disponible qu'en États-Unis.

Création d'un type de slot composite

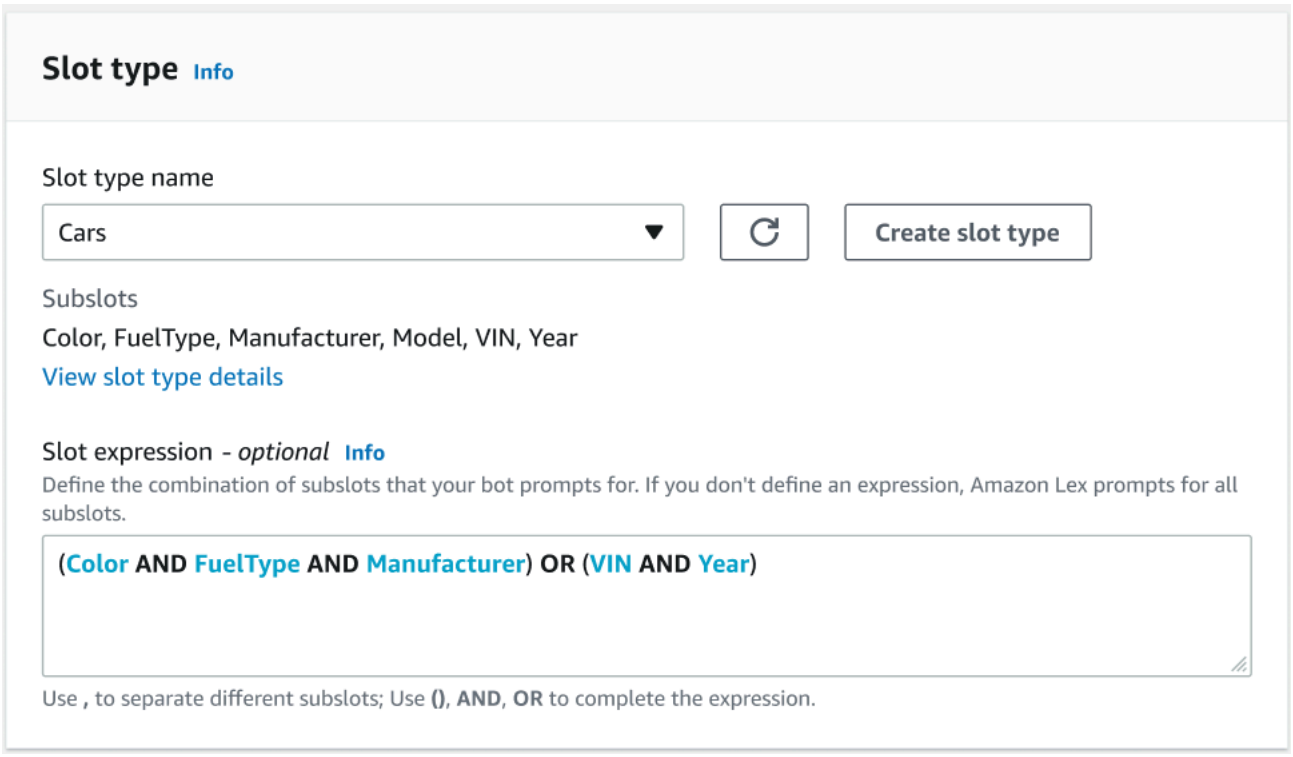
Pour utiliser des sous-emplacements dans un emplacement composite, vous devez d'abord configurer le type d'emplacement composite. Pour ce faire, utilisez les étapes de console d'ajout d'un type de slot ou l'opération API. Après avoir choisi le nom et la description du type d'emplacement composite, vous devez fournir des informations sur les sous-emplacements. Pour plus d'informations sur l'ajout d'un type d'emplacement, voir Ajouter des types de slots

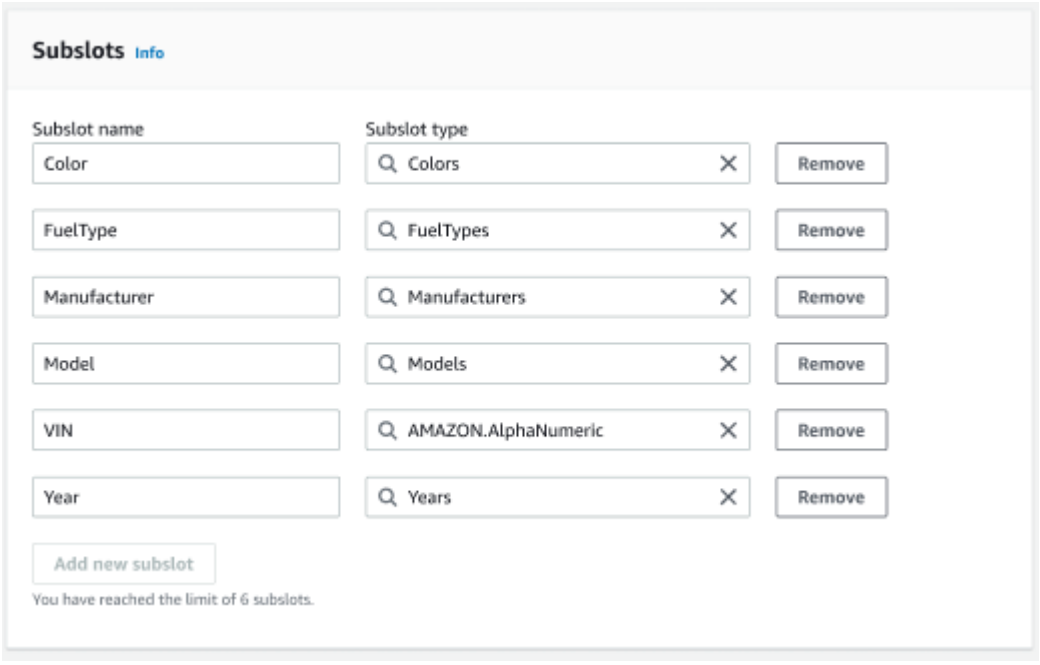
Sous-emplacements

Un type d'emplacement composite nécessite la configuration des emplacements sous-jacents, appelés sous-emplacements. Si vous souhaitez obtenir plusieurs informations de la part d'un client dans le cadre d'une seule demande, configurez une combinaison de sous-emplacements. Par exemple : ville, État et code postal. Vous pouvez ajouter jusqu'à 6 sous-emplacements pour un emplacement composite.

Des créneaux de types de slots singuliers peuvent être utilisés pour ajouter des sous-slots au type de slot composite. Toutefois, vous ne pouvez pas utiliser un type d'emplacement composite comme type d'emplacement pour un sous-emplacement.

Les images suivantes illustrent un emplacement composite « Car », qui est une combinaison de sous-emplacements : couleurFuelType, fabricant, modèle, VIN et année.





Générateur d'expressions

Pour gérer l'exécution d'un slot composite, vous pouvez éventuellement utiliser le générateur d'expressions. Avec le générateur d'expressions, vous pouvez concevoir une expression d'emplacement logique pour capturer les valeurs de sous-emplacement requises dans l'ordre souhaité. Dans le cadre de l'expression booléenne, vous pouvez utiliser des opérateurs tels que AND

et OR. Sur la base de l'expression conçue, lorsque les sous-créneaux requis sont remplis, le créneau composite est considéré comme rempli.

Utilisation d'un type de fente composite

Dans certains cas, vous souhaiterez peut-être capturer différents emplacements dans le cadre d'un seul emplacement. Par exemple, un robot de planification de l'entretien d'un véhicule peut avoir l'intention suivante :

```
My car is a {car}
```

L'intention est que l'emplacement composite {car} contienne une liste des emplacements, comprenant des détails sur le véhicule. Par exemple, « Toyota Camry blanche 2021 ».

Le slot composite diffère d'un slot à valeurs multiples. L'emplacement composite est composé de plusieurs emplacements, chacun ayant sa propre valeur. Alors qu'un slot à valeurs multiples est un slot unique qui peut contenir une liste de valeurs. Pour plus d'informations sur les créneaux à valeurs multiples, voir, Utilisation de plusieurs valeurs dans un emplacement

Pour un emplacement composite, Amazon Lex renvoie une valeur pour chaque sous-emplacement en réponse à l'`RecognizeUtterance`opération `RecognizeText` or. Les informations relatives à l'emplacement renvoyées pour l'énoncé sont les suivantes : « Je souhaite planifier un entretien pour ma « Toyota Camry blanche 2021 » auprès du bot. CarService

```
"slots": {
    "CarType": {
        "value": {
            "originalValue": "White Toyota Camry 2021",
            "interpretedValue": "White Toyota Camry 2021",
            "resolvedValues": [
                "white Toyota Camry 2021"
            ]
        },
        "subSlots": {
            "Color": {
                "value": {
                    "originalValue": "White",
                    "interpretedValue": "White",
                    "resolvedValues": [
                        "white"
                    ]
                },
```

```
                "shape": "Scalar"
            },
            "Manufacturer": {
                "value": {
                    "originalValue": "Toyota",
                    "interpretedValue": "Toyota",
                    "resolvedValues": [
                        "Toyota"
                    ]
                },
                "shape": "Scalar"
            },
            "Model": {
                "value": {
                    "originalValue": "Camry",
                    "interpretedValue": "Camry",
                    "resolvedValues": [
                       "Camry"
                    ]
                },
                "shape": "Scalar"
            },
            "Year": {
                "value": {
                    "originalValue": "2021",
                    "interpretedValue": "2021",
                    "resolvedValues": [
                        "2021"
                    ]
                },
                "shape": "Scalar"
            }
        }
    },
    ...
}
```

Un créneau composite peut être obtenu au premier tour ou au n-e tour d'une conversation. Sur la base des valeurs d'entrée fournies, le slot composite peut obtenir les sous-slots restants requis.

Les emplacements composites renvoient toujours une valeur pour chaque sous-emplacement. Lorsque l'énoncé ne contient pas de valeur reconnaissable pour un sous-emplacement donné, aucune réponse n'est renvoyée pour ce sous-emplacement particulier.

Les emplacements composites fonctionnent à la fois avec la saisie de texte et la saisie vocale.

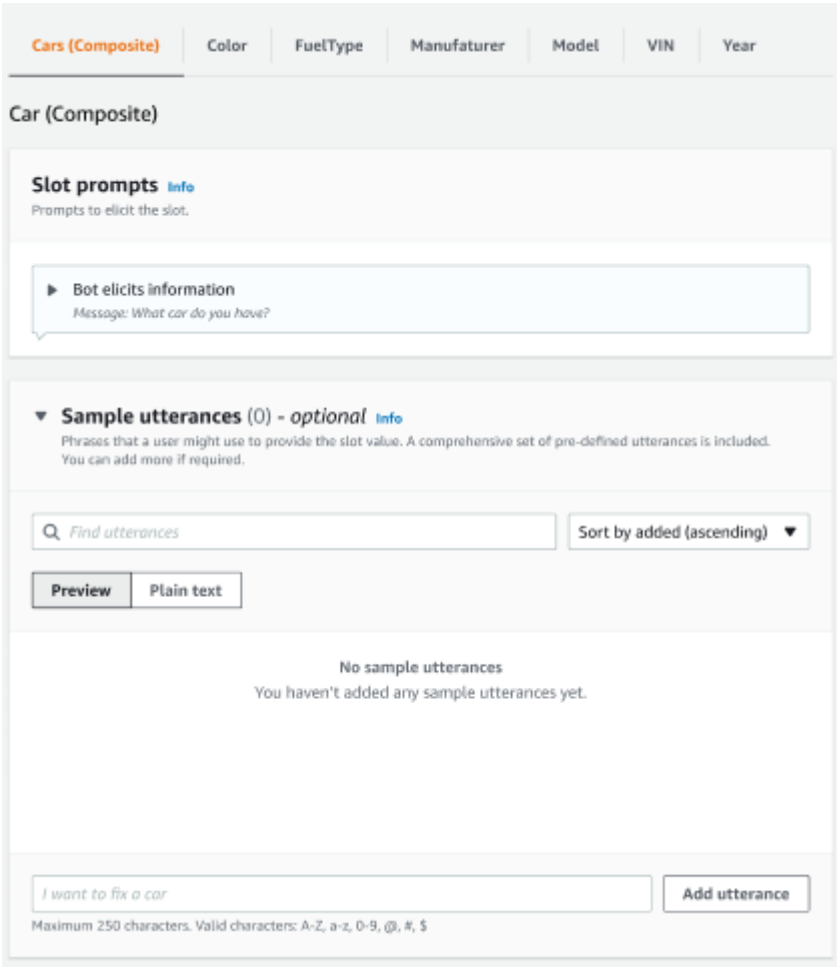
Lorsque vous ajoutez un emplacement à une intention, un emplacement composite n'est disponible qu'en tant que type d'emplacement personnalisé.

Vous pouvez utiliser des emplacements composites dans les invites. Par exemple, vous pouvez définir l'invite de confirmation pour une intention.

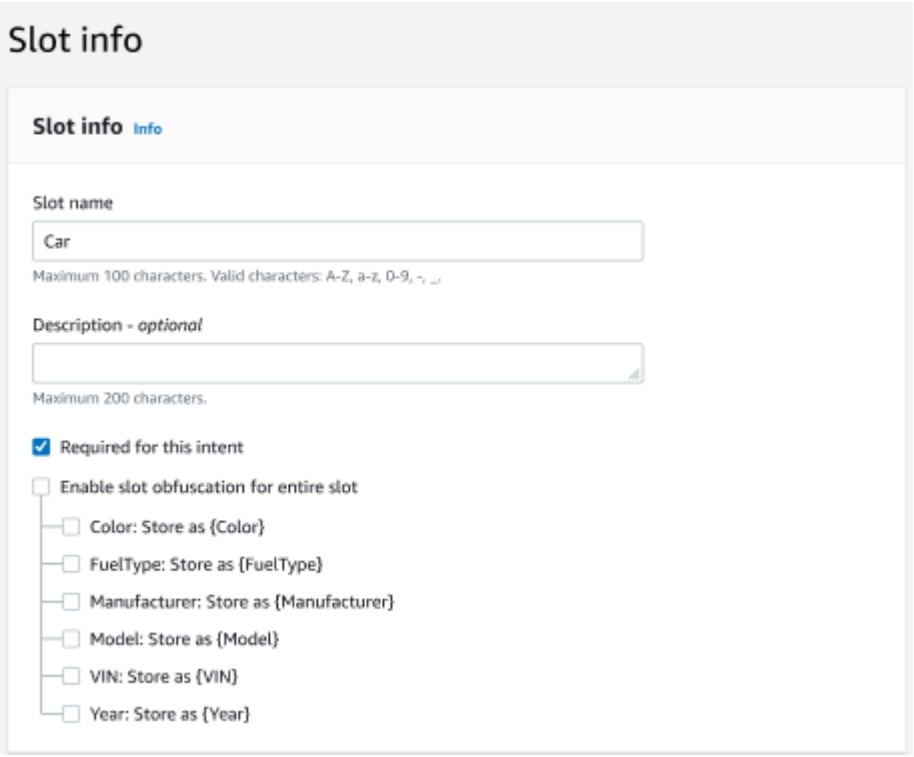
```
Would you like me to schedule service for your 2021 White Toyota Camry?
```

Lorsqu'Amazon Lex envoie le message à l'utilisateur, celui-ci envoie le message suivant :
« Souhaitez-vous que je planifie l'entretien de votre Toyota Camry blanche 2021 ? »

Chaque sous-emplacement est configuré comme un emplacement. Vous pouvez ajouter des instructions pour obtenir le sous-créneau et des exemples d'énoncés. Vous pouvez activer la fonction d'attente et de poursuite pour un sous-emplacement ainsi que les valeurs par défaut. Pour de plus amples informations, consultez Utilisation des valeurs d'emplacement par défaut.



Vous pouvez utiliser l'obfuscation des emplacements pour masquer l'ensemble de l'emplacement composite dans les journaux de conversation. Veuillez noter que l'obfuscation des emplacements est appliquée au niveau de l'emplacement composite et que lorsqu'elle est activée, les valeurs des sous-emplacements appartenant à un emplacement composite sont masquées. Lorsque vous masquez des valeurs d'emplacement, la valeur de chacune des valeurs d'emplacement est remplacée par le nom de l'emplacement. Pour plus d'informations, veuillez consulter Masquer les valeurs des créneaux dans les journaux de conversation.



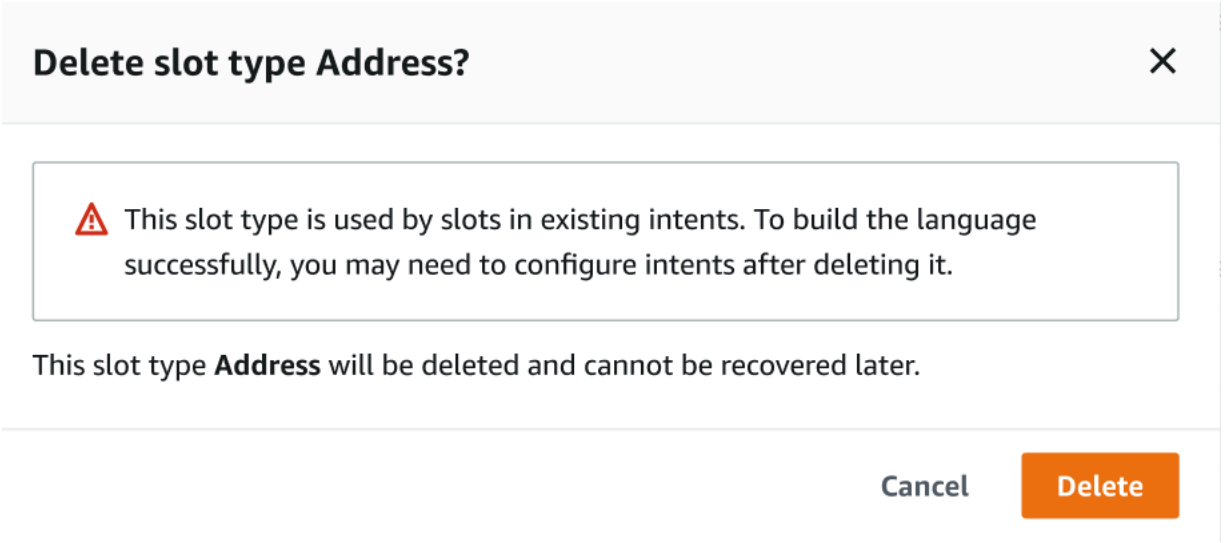
Modification d'un type de slot composite

Vous pouvez modifier un sous-emplacement à partir de la configuration composite du slot afin de modifier le nom du sous-slot et le type de slot. Toutefois, lorsqu'un emplacement composite est utilisé par une intention, vous devez modifier les intentions avant de modifier le sous-emplacement.

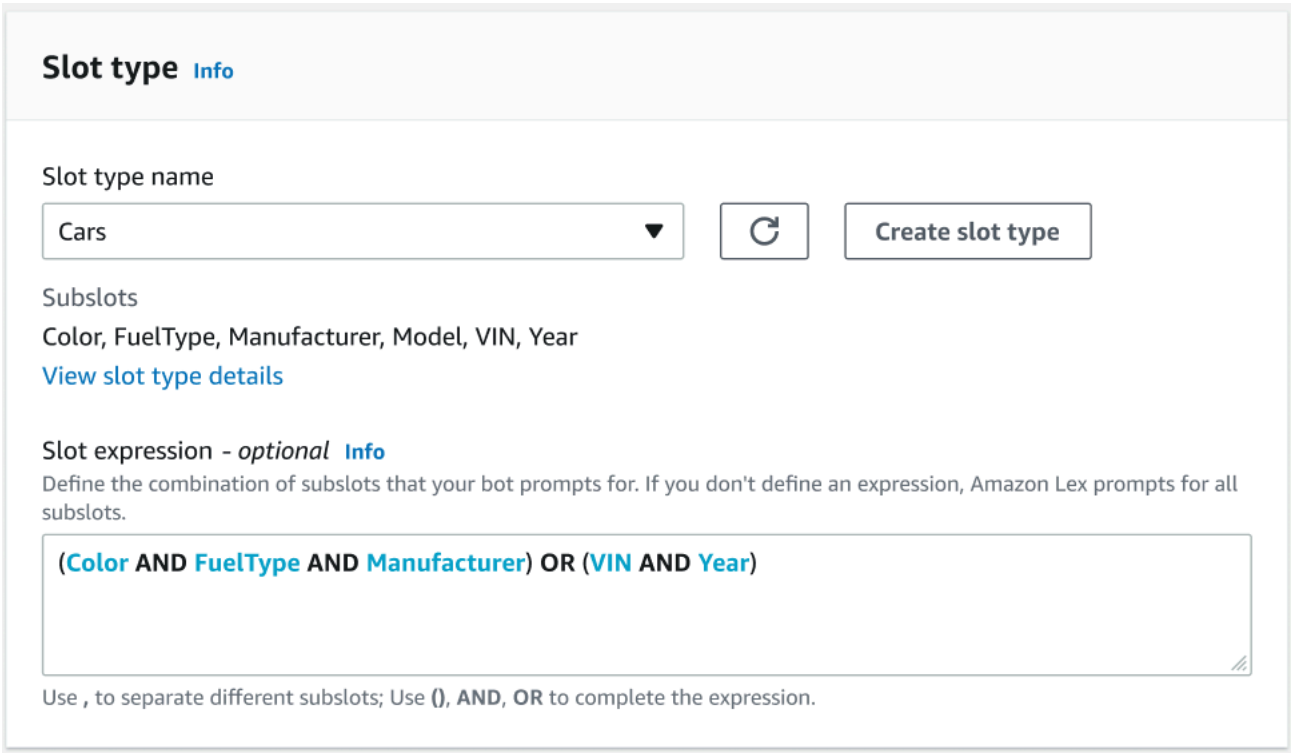
> Existing intents use this slot type. To build the language successfully, you may need to configure those intents after editing sub slots.

Supprimer un type de slot composite

Vous pouvez supprimer un sous-emplacement depuis la configuration du slot composite. Veuillez noter que lorsqu'un sous-emplacement est utilisé dans le cadre d'une intention, les sous-emplacements sont toujours supprimés de cette intention.



L'expression d'emplacement dans le générateur d'expressions fournit une alerte pour informer sur les sous-emplacements supprimés.



Tester un bot à l'aide de la console

La console Amazon Lex V2 contient une fenêtre de test que vous pouvez utiliser pour tester l'interaction avec votre bot. Vous utilisez la fenêtre de test pour avoir une conversation test avec votre bot et pour voir les réponses que votre application reçoit du bot.

Il existe deux types de tests que vous pouvez effectuer avec votre bot. Le premier, le test express, vous permet de tester votre bot avec les phrases exactes que vous avez utilisées pour le créer. Par exemple, si vous avez ajouté l'énoncé « Je veux cueillir des fleurs » à votre intention, vous pouvez tester le bot en utilisant cette phrase exacte.

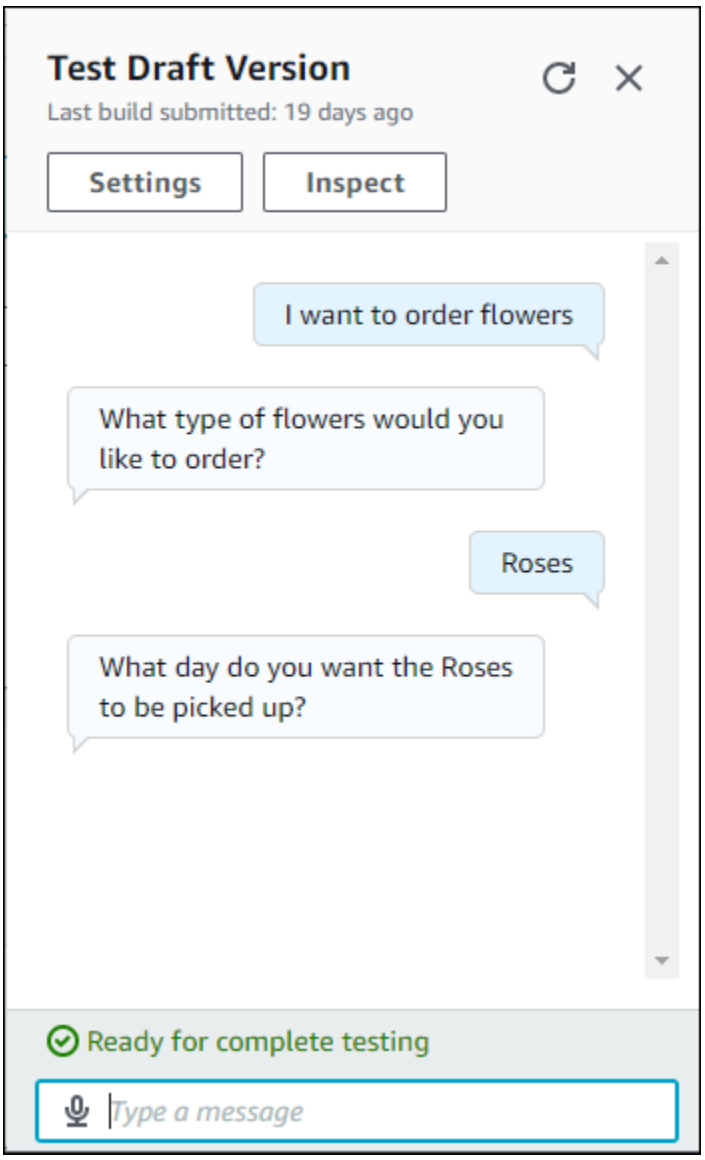
Le second type, le test complet, vous permet de tester votre bot à l'aide de phrases liées aux énoncés que vous avez configurés. Par exemple, vous pouvez utiliser la phrase « Puis-je commander des fleurs » pour démarrer une conversation avec votre bot.

Vous testez un bot à l'aide d'un alias et d'une langue spécifiques. Si vous testez la version de développement du bot, vous utilisez l'`TestBotAlias`alias pour les tests.

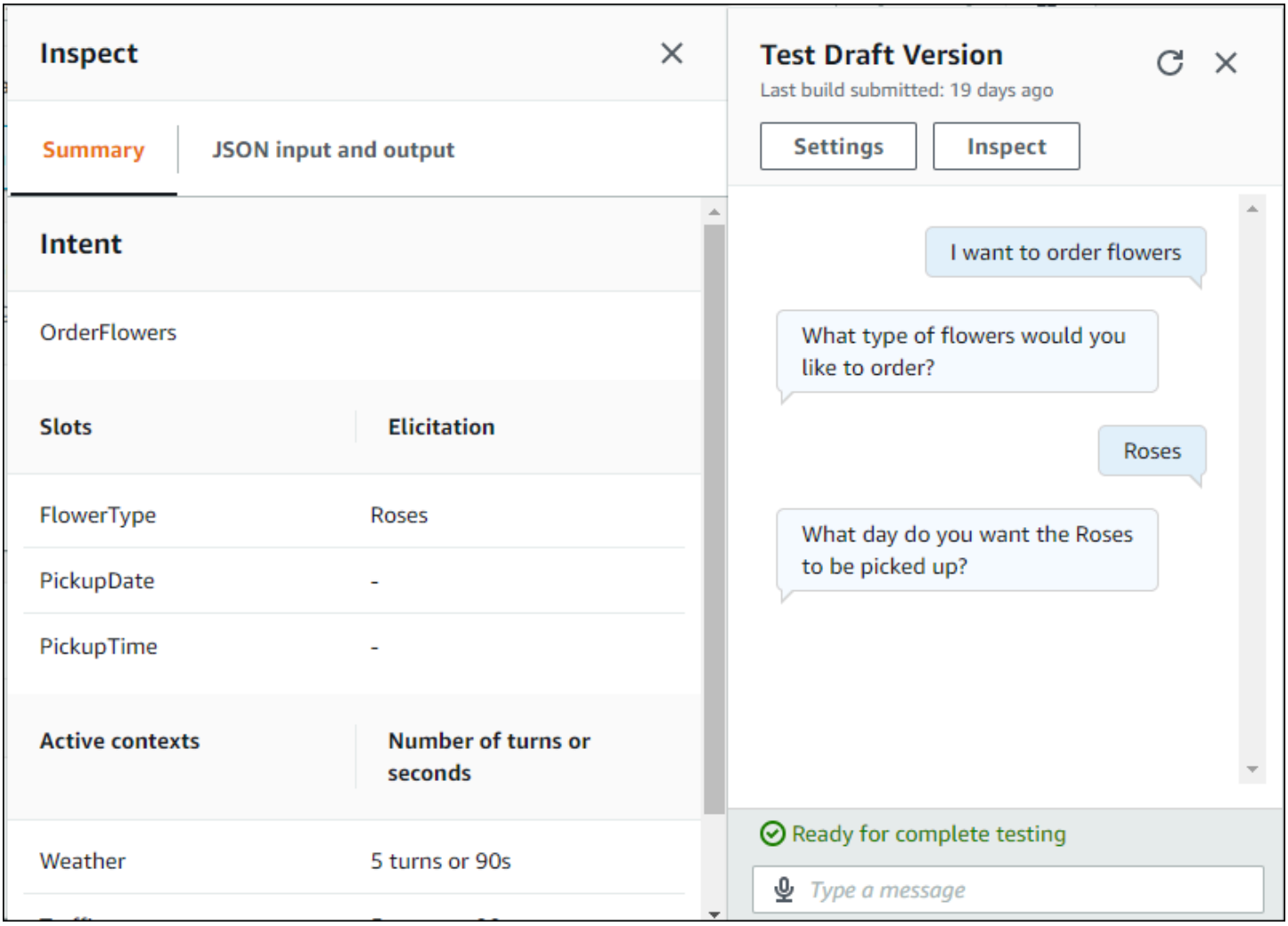
Pour ouvrir la fenêtre de test

1. Connectez-vous à la console Amazon Lex AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/lex/.
2. Choisissez le bot à tester dans la liste des robots.
3. Dans le menu de gauche, choisissez Alias.
4. Dans la liste des alias, choisissez l'alias à tester.
5. Dans Langues, cliquez sur le bouton radio de la langue à tester, puis choisissez Tester.

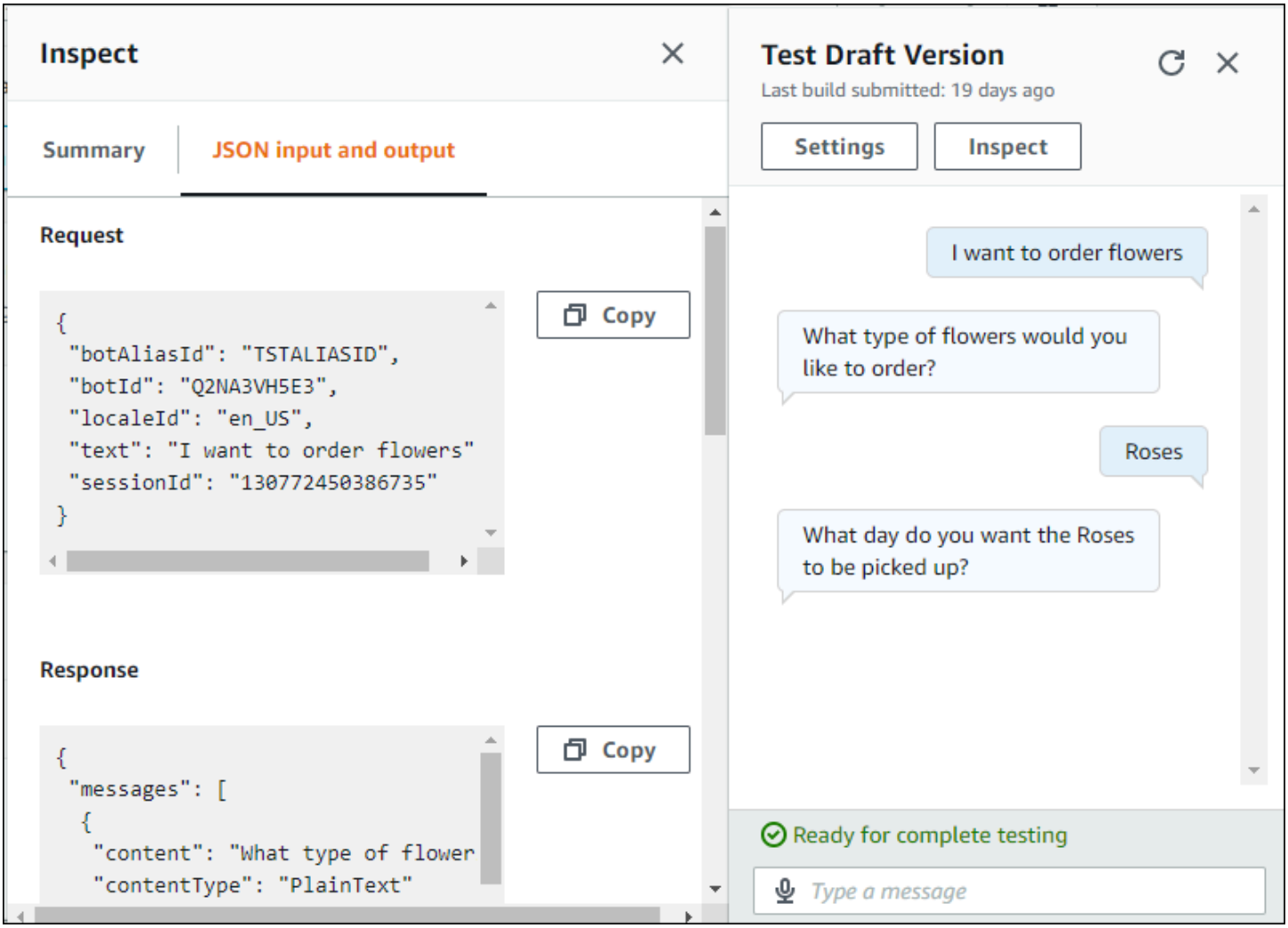
Une fois que vous avez choisi Test, la fenêtre de test s'ouvre dans la console. Vous pouvez utiliser la fenêtre de test pour interagir avec votre bot, comme le montre le graphique suivant.



En plus de la conversation, vous pouvez également sélectionner Inspecter dans la fenêtre de test pour voir les réponses renvoyées par le bot. La première vue affiche un résumé des informations renvoyées par votre bot à la fenêtre de test.



Vous pouvez également utiliser la fenêtre d'inspection des tests pour voir les structures JSON envoyées entre le bot et la fenêtre de test. Vous pouvez voir à la fois la demande depuis la fenêtre de test et la réponse depuis Amazon Lex V2.


Inspect
Summary
JSON input and output
Request
Copy
{
"botAliasId": "TSTALIASID",
"botId": "Q2NA3VH5E3",
"localeId": "en_US",
"text": "I want to order flowers"
"sessionId": "130772450386735"
}
Response
Copy
{
"messages": [
{
"content": "What type of flower
"contentType": "PlainText"
Test Draft Version
Last build submitted: 19 days ago
Settings
Inspect
I want to order flowers
What type of flowers would you like to order?
Roses
What day do you want the Roses to be picked up?
Ready for complete testing
Type a message

Optimisez la création et les performances des robots grâce à l'IA générative

> **Note**
>
> Ces fonctionnalités utilisent l'IA générative. Lorsque vous utilisez le service, n'oubliez pas qu'il peut donner des réponses inexactes ou inappropriées. Pour plus d'informations, consultez la politique d'AWS en matière d'IA responsable.
> Propulsé par Amazon Bedrock : AWS implémente la détection automatique des abus. Les fonctionnalités d'IA générative d'Amazon Lex V2 étant basées sur Amazon Bedrock, les utilisateurs héritent des contrôles mis en œuvre dans Amazon Bedrock pour renforcer la sûreté, la sécurité et l'utilisation responsable de l'IA.

Tirez parti des capacités d'intelligence artificielle générative d'Amazon Bedrock pour automatiser et accélérer le processus de création de votre bot Amazon Lex V2. Vous pouvez effectuer les processus suivants à l'aide d'Amazon Bedrock.

- Créez de nouveaux robots et remplissez-les efficacement avec des intentions et des types d'emplacements pertinents à l'aide d'une description en langage naturel.
- Générez automatiquement des exemples d'énoncés pour les besoins de votre robot.
- Améliorez les performances de résolution des machines à sous de vos robots.
- Créez une intention pour répondre aux questions de vos clients.

Vous pouvez activer les fonctionnalités d'IA générative pour Amazon Lex V2 via la console ou l'API.

> **Note**
>
> Avant de pouvoir tirer parti des fonctionnalités de l'IA générative, vous devez remplir les conditions préalables suivantes :
>
> 1. Accédez à la console Amazon Bedrock et inscrivez-vous pour accéder au modèle Anthropic Claude que vous souhaitez utiliser (pour plus d'informations, voir Accès au modèle). Pour plus d'informations sur les tarifs d'utilisation d'Amazon Bedrock, consultez les tarifs d'Amazon Bedrock.

2. Activez les fonctionnalités d'IA générative pour les paramètres régionaux de votre bot. Pour ce faire, suivez les étapes indiquées surOptimisez la création et les performances des robots grâce à l'IA générative.

Using the console

1. Connectez-vous à la console Amazon Lex V2 AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/lexv2/home.
2. Sélectionnez le bot et les paramètres régionaux du bot pour lesquels vous souhaitez activer les fonctionnalités d'IA générative.
3. Dans la section Configurations génératives de l'IA, sélectionnez Configurer.
4. Activez le bouton Activé pour chaque fonctionnalité que vous souhaitez activer. Sélectionnez le modèle et la version que vous souhaitez utiliser pour cette fonctionnalité. L'activation d'une fonctionnalité peut entraîner des frais supplémentaires. Pour plus d'informations sur les tarifs d'utilisation d'Amazon Bedrock, consultez les tarifs d'Amazon Bedrock. Pour en savoir plus sur une fonctionnalité, sélectionnez le sujet correspondant dans la liste ci-dessous. Sélectionnez Enregistrer après avoir activé les fonctionnalités que vous souhaitez activer. Une bannière verte de réussite apparaît pour confirmer que les fonctionnalités sont activées.

Using the API

1. Pour activer les fonctionnalités d'IA générative pour un nouveau bot, utilisez l'CreateBotopération pour créer un nouveau bot.
2. Envoyez une CreateBotLocaledemande en modifiant l'`generativeAISettings`objet si nécessaire. Si vous activez les fonctionnalités d'un bot existant, envoyez plutôt une UpdateBotLocaledemande.

    a. Pour activer l'utilisation du générateur de robots descriptifs, modifiez l'`descriptiveBotBuilder`objet. Spécifiez le modèle de base à utiliser `modelArn` sur le terrain et définissez la `enabled` valeur sur`True`.

    b. Pour activer l'amélioration de la résolution des créneaux, modifiez l'`slotResolutionImprovement`objet. Spécifiez le modèle de base à utiliser `modelArn` sur le terrain et définissez la `enabled` valeur sur`True`.

c. Pour activer la génération d'exemples d'énoncés, modifiez l'`sampleUtteranceGeneration`objet. Spécifiez le modèle de base à utiliser `modelArn` sur le terrain et définissez la `enabled` valeur sur`True`.

Rubriques

Utilisation du générateur de bots descriptif

> **Note**
>
> Avant de pouvoir tirer parti des fonctionnalités de l'IA générative, vous devez remplir les conditions préalables suivantes
>
> 1. Accédez à la console Amazon Bedrock et inscrivez-vous pour accéder au modèle Anthropic Claude que vous souhaitez utiliser (pour plus d'informations, voir Accès au modèle). Pour plus d'informations sur les tarifs d'utilisation d'Amazon Bedrock, consultez les tarifs d'Amazon Bedrock.
> 2. Activez les fonctionnalités d'IA générative pour les paramètres régionaux de votre bot. Pour ce faire, suivez les étapes indiquées surOptimisez la création et les performances des robots grâce à l'IA générative.

Le générateur de bots descriptif vous permet de tirer parti de l'accès d'Amazon Bedrock à de grands modèles linguistiques pour améliorer l'efficacité du processus de création de robots. Vous fournissez une invite en langage naturel qui inclut l'objectif du bot et les actions qu'il doit effectuer. Amazon Lex V2 exploite les capacités d'Amazon Bedrock pour générer des intentions et des types d'emplacements pertinents pour votre bot en fonction de votre description. Une fois que vous avez choisi les intentions et les types d'emplacements que vous souhaitez conserver, vous pouvez ensuite itérer sur le bot pour l'adapter à votre cas d'utilisation spécifique. Le générateur de bot descriptif vous fait gagner du temps en vous évitant d'avoir à créer manuellement des intentions et des types d'emplacements pour le bot.

Le générateur de bots descriptif est disponible dans les paramètres régionaux anglais (voir les paramètres régionaux commençant par en_ dans le tableau ci-dessousLangues et paramètres régionaux pris en charge par Amazon Lex V2).

Avant de créer votre bot, procédez comme suit.

1. Vérifiez que votre rôle dispose des autorisations appropriées en consultant les étapes indiquées surAutorisations nécessaires pour créer un bot avec une description en langage naturel.
2. Décidez de la description à utiliser. Vous pouvez vous référer à Exemples de descriptions de robots des exemples de descriptions de robots.

Créez un bot en utilisant un langage naturel pour décrire ce que le bot devrait être capable de faire. Amazon Lex V2 invoque les modèles Amazon Bedrock pour générer des intentions et des types d'emplacements adaptés au cas d'utilisation de votre bot. Vous pouvez créer le bot à l'aide de la console ou de l'API.

Console

Créez un bot à l'aide du générateur de bot descriptif

1. Connectez-vous à la console Amazon Lex V2 AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/lexv2/home.
2. Sur la page Bots, sélectionnez Créer un bot.
3. Pour la méthode de création, choisissez Descriptive Bot Builder - GenAI.
4. Donnez un nom et une description facultative à votre bot, configurez les autorisations IAM et choisissez si votre bot est soumis à la COPPA ou non. Sélectionnez ensuite Next.
5. Sélectionnez une langue dans laquelle créer le bot, une voix pour le bot et un seuil de confiance pour la classification des intentions (pour plus d'informations, voirUtilisation des scores de confiance en matière d'intention.
6. Sous Descriptive Bot Builder - GenAI, décrivez le bot que vous souhaitez créer. Votre description doit être à la fois détaillée et précise afin de générer des intentions appropriées et suffisantes pour votre bot. Incluez une liste d'actions visant à améliorer le processus de création d'intentions.
7. Sélectionnez un fournisseur de modèles et un modèle sous Sélectionner un modèle.
8. Pour créer le bot dans un autre environnement régional, choisissez Ajouter une autre langue. Lorsque vous avez terminé d'ajouter des langues, sélectionnez OK. Amazon Lex V2 crée

votre bot et le générateur de bot descriptif génère des intentions et des emplacements pour celui-ci. Lorsque les paramètres régionaux ont été générés, la bannière passe du bleu au vert. Sélectionnez Vérifier pour voir les intentions et les types d'emplacements générés.

> **Note**
>
> Le générateur de bots descriptif n'est actuellement disponible que dans les langues anglaises. Toutefois, vous pouvez copier un bot dans une langue autre que l'anglais après l'avoir créé.

Passez en revue les intentions et les types d'emplacements générés et ajoutez-les à votre bot

1. S'il existe suffisamment d'intentions et de types d'emplacements applicables au cas d'utilisation de votre bot, vous pouvez consulter les intentions générées.

   a. Passez en revue les intentions générées.

      i. Cochez une case à côté d'une intention pour la supprimer de la liste des intentions à ajouter au bot.

      ii. Choisissez un nom d'intention pour afficher les exemples d'énoncés et les emplacements générés pour l'intention.

      iii. Par défaut, tous les énoncés et tous les emplacements sont sélectionnés. Cochez une case pour supprimer cet élément de l'intention. Sélectionnez Ajouter à la sélection pour que les éléments cochés restent conformes à l'intention.

   b. Passez en revue les types d'emplacements générés.

      i. Cochez une case à côté d'un type d'emplacement pour le supprimer de la liste des intentions à ajouter au bot.

      ii. Vous pouvez ajouter des valeurs à un type de slot après l'avoir ajouté au bot.

2. Lorsque vous êtes satisfait de vos intentions et de vos types d'emplacements, sélectionnez Ajouter des intentions et des types d'emplacements en haut de la page pour ajouter les intentions et les types d'emplacements à votre bot.

3. Lorsque les ressources ont fini d'être ajoutées, une bannière verte de réussite apparaît. Accédez à Intentions et types d'emplacements pour modifier ceux qui ont été générés et pour ajouter des valeurs supplémentaires.

4. Si les types Generated intents et Generated slot sont généralement inapplicables au bot que vous souhaitez créer, effectuez les étapes suivantes.

   a. Sélectionnez Nouvelle génération dans la section Détails du générateur de bots descriptif.

   b. Réécrivez l'invite et sélectionnez Régénérer pour générer de nouvelles intentions et de nouveaux types d'emplacements. Les résultats sont différents si vous utilisez un autre modèle.

   > **Important**
   >
   > Rien ne garantit que les mêmes intentions et les mêmes créneaux seront générés. Vous êtes débité chaque fois que vous régénérez les intentions et les types de créneaux.

API

Créez le bot en utilisant une description en langage naturel

Lorsque vous utilisez le générateur de bot descriptif via l'API, il crée une définition de bot dans un fichier .zip d'un compartiment Amazon S3. Vous téléchargez ce fichier et importez la définition du bot dans Amazon Lex V2 pour créer votre bot.

1. Envoyez une CreateBotdemande pour créer un nouveau bot. Envoyez ensuite une CreateBotLocaledemande pour créer une localisation pour le bot.
2. Envoyez une StartBotResourceGenerationdemande en spécifiant l'ID, la version et les paramètres régionaux du bot. Vous pouvez utiliser `DRAFT` pour la version bot. Indiquez votre invite sur le `generationInputPrompt` terrain. Votre description doit être à la fois détaillée et précise afin de générer des intentions appropriées et suffisantes pour votre bot. Incluez une liste d'actions visant à améliorer le processus de création d'intentions.
3. Prenez note de cela `generationId` dans la réponse.
4. Envoyez une DescribeBotResourceGenerationdemande en utilisant le code `generationId` que vous avez reçu dans la `StartBotResourceGeneration` réponse. Incluez l'ID, la version et les paramètres régionaux du bot.
5. Si la `DescribeBotResourceGeneration` réponse est « `generationStatus` in »`Complete`, le `generatedBotLocaleUrl` champ sera

également renseigné. Utilisez cet URI Amazon S3 pour télécharger la définition du bot en suivant les étapes de la section Téléchargement d'un objet.

Vérifiez la définition du bot générée et importez-la

1. Utilisez l'URI Amazon S3 figurant `generationStatus` dans la `DescribeBotResourceGeneration` réponse pour télécharger la définition du bot en suivant les étapes de la section Téléchargement d'un objet.
2. Vous pouvez modifier directement le contenu généré pour le cas d'utilisation spécifique de votre bot en modifiant le fichier. Vous pouvez également envoyer une autre `StartBotResourceGeneration` demande pour régénérer les intentions et les emplacements.

   > **Important**
   >
   > Rien ne garantit que les mêmes intentions et les mêmes créneaux seront générés. Vous êtes débité chaque fois que vous régénérez les intentions et les types de créneaux.

3. Pour importer la définition du bot, suivez les étapes décrites dansImportation.
4. Après l'importation, vous pouvez modifier les intentions et les emplacements générés à l'aide des UpdateSlotTypeopérations UpdateIntentUpdateSlot, et.

Pour répertorier les métadonnées relatives à tous les éléments générés pour les paramètres régionaux d'un bot, utilisez l'ListBotResourceGenerationsopération. Utilisez l'une des `generationId` valeurs renvoyées dans une `DescribeBotResourcGeneration` demande pour récupérer l'URI Amazon S3 pour une définition de bot générée.

Rubriques

- Exemples de descriptions de robots
- Autorisations nécessaires pour créer un bot avec une description en langage naturel

Exemples de descriptions de robots

Industry	Exemple d'invite
Services financiers	Nous sommes un service de cartes financières qui aide les utilisateurs à effectuer des tâches lorsqu'ils reçoivent une nouvelle carte, telles que l'activation de la carte, l'envoi d'un code PIN par e-mail ou par la poste, la vérification d'une nouvelle carte (à l'aide d'un code postal). Nous les aidons également dans les tâches associées à leur carte existante, telles que se renseigner sur les avantages de la carte de crédit, signaler une carte perdue, demander une nouvelle carte, réinitialiser le code PIN d'une carte ou payer une facture de carte.
Services de restauration	Je veux qu'un robot aide les clients à commander de la nourriture (en utilisant le numéro d'article, la quantité, la taille), à vérifier le statut de la commande et à annuler une commande. Utilisez le numéro de commande pour indexer les commandes.
Compagnie aérienne	Nous sommes un domaine de compagnies aériennes qui aide les utilisateurs à réserver des billets d'avion, à vérifier les détails d'une réservation, à obtenir un reçu pour un vol réservé, à se renseigner sur le statut du vol, à reprogrammer les vols réservés, à obtenir les détails des vols et à annuler les vols réservés. Vous pouvez également générer des intentions supplémentaires si elles contribuent à soutenir les fonctions de la description du domaine.
Assurance	Objectif : Nous sommes une compagnie d'assurance qui vend des polices d'assuran

Industry	Exemple d'invite
	ce automobile, habitation et rente. Je veux un robot capable de vérifier l'état d'une réclamati on, de déposer une réclamation, d'effectuer des paiements de police et d'annuler une police. Nous utilisons policy_id et les 4 derniers du SSN pour l'identification et la validation du compte. Je pense que le bot aura au moins les objectifs et emplacements suivants : authentic ation - policy_id, last4SSNType de politique : voiture, maison, annuityPolicy status : vérifier le solde, vérifier la date d'échéance, vérifier la couvertureEffectuer un paiement : paiement unique, versements échelonnés, montant
Gestion des véhicules	Nous sommes en train de créer un robot de recherche de voitures remorquées qui aide les conducteurs d'une ville dont la voiture a été remorquée à trouver où se trouve la voiture. Ce robot doit demander l'adresse ou l'emplace ment d'où l'automobile a été remorquée, ainsi que des détails sur le véhicule, tels que la plaque d'immatriculation, la marque, le modèle et l'année de la voiture. Le bot doit répondre en indiquant l'emplacement du parking remorqué et les heures d'ouverture.

Industry	Exemple d'invite
Voyage	Je suis agent de voyages et je souhaite qu'un robot aide mes clients à réserver un voyage à Disney. Disney propose plusieurs parcs dans le monde entier, ainsi que des hôtels, des restaurants et des divertissements spéciaux qui peuvent être réservés. Les utilisateurs du bot devraient pouvoir modifier ou annuler leur réservation. Les réservations doivent inclure au minimum le parc, les dates et l'hôtel. L'inclusion des repas ou des divertissements est facultati ve et peut être ajoutée ou modifiée ultérieur ement.

Autorisations nécessaires pour créer un bot avec une description en langage naturel

- Pour accéder à cette fonctionnalité sur la console Amazon Lex V2, assurez-vous que votre rôle de console est `bedrock:ListFoundationModels` autorisé.
- Le rôle IAM associé au bot doit être `bedrock:InvokeModel` autorisé. Lorsque vous activez cette fonctionnalité avec la console Amazon Lex, la politique est automatiquement ajoutée au rôle de bot, à condition que votre bot utilise un rôle lié à un service généré par Amazon Lex.

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [
              "bedrock:InvokeModel"
          ],
          "Resource": [
              "arn:aws:bedrock:region::foundation-model/model-id"
          ]
      }
  ]
}
```

Génération d'énoncés

> **Note**
>
> Avant de pouvoir tirer parti des fonctionnalités de l'IA générative, vous devez remplir les conditions préalables suivantes
>
> 1. Accédez à la console Amazon Bedrock et inscrivez-vous pour accéder au modèle Anthropic Claude que vous souhaitez utiliser (pour plus d'informations, voir Accès au modèle). Pour plus d'informations sur les tarifs d'utilisation d'Amazon Bedrock, consultez les tarifs d'Amazon Bedrock.
> 2. Activez les fonctionnalités d'IA générative pour les paramètres régionaux de votre bot. Pour ce faire, suivez les étapes indiquées surOptimisez la création et les performances des robots grâce à l'IA générative.

Utilisez la génération d'énoncés pour automatiser la création d'exemples d'énoncés adaptés à vos besoins. Au lieu de saisir manuellement des exemples d'énoncés, Amazon Lex V2 génère des exemples d'énoncés pour vous en fonction du nom de l'intention, de la description et des exemples d'énoncés existants, afin que vous puissiez réduire le temps et les efforts que vous consacrez à la découverte et à la rédaction de vos propres exemples d'énoncés. Une fois qu'Amazon Lex V2 a généré des énoncés, vous pouvez les modifier et les supprimer. Utilisez cet outil pour accélérer la création d'exemples d'énoncés pour le processus de reconnaissance d'intention.

Pour autoriser la génération d'énoncés, suivez les étapes décrites dans la section pour activer les Optimisez la création et les performances des robots grâce à l'IA générative fonctionnalités d'IA générative.

Vous pouvez générer des énoncés à l'aide de la console ou de l'API.

Console

1. Accédez à la section Exemples d'énoncés de n'importe quelle intention de votre bot (dans le générateur de conversation visuel, elle se trouve dans le bloc Démarrer).

2. Cliquez sur le bouton Générer des énoncés pour générer 5 exemples d'énoncés. Si votre intention comporte plus de 25 exemples d'énoncés, le bouton Générer des énoncés est désactivé.
3. Les énoncés générés sont affichés avec une bannière verte qui différencie les énoncés générés des énoncés existants.
4. Passez le curseur sur un énoncé pour afficher les options permettant de modifier, de supprimer et de trier les énoncés générés.

API

1. Envoyez une GenerateBotElementdemande en indiquant l'intention, l'ID du bot, la version et les paramètres régionaux pour lesquels vous souhaitez générer des exemples d'énoncés.
2. La réponse renvoie une liste d' SampleUtteranceobjets contenant chacun un énoncé généré.
3. Pour ajouter les énoncés à l'intention, envoyez une UpdateIntentdemande et ajoutez les énoncés dans le champ. `sampleUtterances`

Rubriques

- Autorisations pour la génération d'énoncés

Autorisations pour la génération d'énoncés

Pour accéder à cette fonctionnalité sur la console Amazon Lex V2, assurez-vous que votre rôle de console dispose `bedrock:ListFoundationModels` des `bedrock:InvokeModel` autorisations nécessaires.

Utilisation de la résolution des créneaux assistée

> **Note**
>
> Avant de pouvoir tirer parti des fonctionnalités de l'IA générative, vous devez remplir les conditions préalables suivantes
>
> 1. Accédez à la console Amazon Bedrock et inscrivez-vous pour accéder au modèle Anthropic Claude que vous souhaitez utiliser (pour plus d'informations, voir Accès au

modèle). Pour plus d'informations sur les tarifs d'utilisation d'Amazon Bedrock, consultez les tarifs d'Amazon Bedrock.

2. Activez les fonctionnalités d'IA générative pour les paramètres régionaux de votre bot. Pour ce faire, suivez les étapes indiquées surOptimisez la création et les performances des robots grâce à l'IA générative.

Vous pouvez améliorer la précision de certains créneaux intégrés dans le flux de conversation de votre bot en utilisant la résolution assistée des créneaux. La résolution assistée des créneaux utilise les grands modèles linguistiques (LLM) d'Amazon Bedrock pour améliorer la reconnaissance de certains créneaux intégrés, ce qui se traduit par une meilleure interprétation des réponses des clients lors de la sélection des créneaux. Pour les énoncés qui n'ont pas pu être résolus normalement, Amazon Lex tentera de les résoudre une deuxième fois à l'aide d'Amazon Bedrock.

La résolution assistée des emplacements vous permet d'utiliser la puissance des modèles de base Amazon Bedrock pour améliorer la précision des emplacements intégrés suivants :

- `AMAZON.Alphanumeric`sans support de regex
- `AMAZON.City`
- `AMAZON.Country`
- `AMAZON.Date`
- `AMAZON.Number`
- `AMAZON.PhoneNumber`
- `AMAZON.Confirmation`

Vous pouvez activer la résolution assistée des créneaux pour toute utilisation utilisant les emplacements intégrés répertoriés ci-dessus. La résolution assistée des emplacements ne s'applique pas aux emplacements personnalisés ou aux emplacements intégrés d'Amazon non répertoriés ci-dessus.

Vous pouvez recueillir des données sur les améliorations de précision après avoir activé la résolution assistée des créneaux dans votre bot Amazon Lex à l'aide des journaux de conversation et des statistiques.

- Journaux de conversation : les interprétations seront présentées `interpretationSource` comme `Bedrock` si Amazon Bedrock avait été utilisé pour résoudre le problème.

- CloudWatch métriques - Les métriques seront publiées sous les dimensions répertoriées dans la CloudWatch métrique. Pour en savoir plus, consultez la section Surveillance d'Amazon Lex avec Amazon CloudWatch.

Pour utiliser le générateur de bot descriptif, assurez-vous que votre rôle IAM dispose des autorisations appropriées en suivant les étapes décrites dansAutorisations pour la résolution assistée des créneaux.

Rubriques

Exemples de résolution assistée par créneaux

Vous trouverez ci-dessous quelques exemples où la résolution assistée par créneaux est capable de transformer intelligemment les énoncés de l'utilisateur en une valeur.

Numéro Amazon

Vertical	Type de machine à sous	slotName	Slot Prompt	énoncé	Valeur résolue
Voyage	Numéro Amazon	numberOfNightsResté	Combien de nuits êtes-vous restée pendant le voyage ?	Une semaine entière, 7 nuits.	7
Services bancaires	Numéro Amazon	numberOfPeopleOnTheAccount	Combien de personnes sont inscrites sur le compte ?	Ma femme et moi.	2

Vertical	Type de machine à sous	slotName	Slot Prompt	énoncé	Valeur résolue
Voyage	Numéro Amazon	numberOfStops	Combien d'arrêts ?	Une fois au Japon. Une fois à Los Angeles.	2

AMAZON. AlphaNumeric

Vertical	Type de machine à sous	slotName	Slot Prompt	énoncé	Valeur résolue
Location de voiture	Amazon. Alphanumérique	ID de transaction	Quel est votre numéro de transaction ?	Je crois que c'était Alpha Whiskey Echo Eight Three Four Nine Romeo Juliet.	AWE8349RJ
Voyage	Amazon. Alphanumérique	Code de confirmation	Quel est le numéro de confirmation de votre réservation ?	Le numéro de confirmation est BLT2UE.	BLT2UE

Amazon Date

Vertical	Type de machine à sous	slotName	Slot Prompt	énoncé	Valeur résolue	Date actuelle
Location de voiture	Amazon Date	Date d'échéance	Quand le contrat de location doit-il expirer ?	Le bail expire le 1er du mois prochain.	2023-12-01	09/11/2023
Voyage	Amazon Date	Date de retour	Quand reviens-tu ?	Plus tard dans la journée, vers 7 heures.	09/11/2023	09/11/2023

AMAZON. PhoneNumber

Vertical	Type de machine à sous	slotName	Slot Prompt	énoncé	Valeur résolue
Assurances	AMAZON. PhoneNumber	Titulaire de la police	Quel est le numéro de téléphone du preneur d'assurance ?	Le numéro de téléphone du titulaire de la police est le 123-456-7890.	1234567890
Vente au détail	AMAZON. PhoneNumber	recherche de téléphone	Quel est ton numéro de téléphone pour que je puisse retrouver ton compte ?	Je pense que c'est sous le 413-570-9617, laissez-moi vérifier.	4135709617

Amazon.country

Vertical	Type de machine à sous	slotName	Slot Prompt	énoncé	Valeur résolue
Voyage	Amazon.country	Pays d'origine	Quel est ton pays d'origine ?	Je suis indienne.	Inde
Services bancaires	Amazon.country	Itinéraire par pays	Dans quels pays allez-vous voyager avec votre carte de débit ?	Je vais me rendre à New Delhi.	Inde

Amazon.City

Vertical	Type d'option	Intention	Question	Réponse	Valeur résolue
Assurances	Amazon.City	policyHolderCity	Dans quelle ville réside le preneur d'assurance ?	J'habite à Springfield.	Springfield
Voyage	Amazon.City	Ville de destination	Dans quelle ville voyagez-vous ?	Je voyage à Tokyo.	Tokyo

Amazon. Confirmation

Vertical	Type de machine à sous	slotName	Slot Prompt	énoncé	Valeur résolue
Assurances	Amazon. Confirmation	Politique expirée	La police d'assurance a-t-elle expiré ?	Oui, malheureusement, il est expiré.	Oui
Services bancaires	Amazon. Confirmation	possède des investissements	Avez-vous des investissements ?	Je n'ai encore investi dans rien.	Non

Activez la résolution assistée des emplacements dans l'écran de configuration de l'IA générative

Vous pouvez activer la résolution assistée pour les emplacements intégrés pris en charge en accédant à l'écran Generative AI.

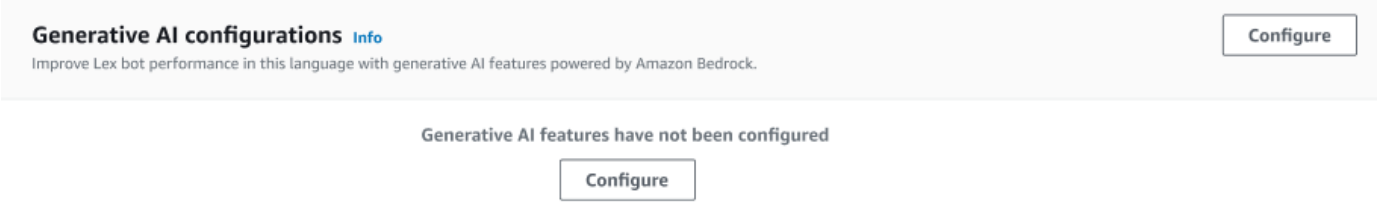
Si le slot est un slot intégré compatible, vous aurez la possibilité d'activer la résolution assistée par slot au niveau du slot.

1. Connectez-vous à la console Amazon Lex V2 AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/lexv2/home.
2. Dans le volet de navigation, sous Bots, sélectionnez le bot que vous souhaitez utiliser pour la résolution assistée des créneaux.
3. Sélectionnez la langue anglais (États-Unis) pour le bot que vous souhaitez activer.
4. Accédez à la section de configuration de l'IA générative à l'écran.
5. Sélectionnez Accéder à Amazon Bedrock pour vous inscrire et activer la fonctionnalité, si elle n'a pas été activée.

> **Note**
>
> Si vous n'avez pas accès aux modèles de fondations Amazon Bedrock, vous devriez voir Go to Amazon Bedrock. Cliquez sur Go to Amazon Bedrock pour accéder à la page Amazon Bedrock où vous pouvez vous inscrire pour accéder aux modèles de

fondation. La résolution assistée des créneaux est actuellement compatible avec Claude V2 et Claude Instant V1. Nous vous conseillons d'utiliser Claude V2 pour de meilleurs résultats.

6. Si vous avez déjà accès aux modèles de Bedrock Foundation, vous devriez voir un bouton Configurer. Cliquez sur ce bouton pour accéder à la page de configuration de l'IA générative afin d'activer les fonctionnalités de l'IA générative dans Lex.



7. Dans le coin supérieur droit de la boîte, déplacez le curseur vers la droite pour sélectionner le paramètre Activé.
8. Cliquez sur le bouton Activer pour activer la résolution assistée des créneaux pour les emplacements sélectionnés.
9. Vous pouvez désactiver la résolution assistée par créneaux en sélectionnant les emplacements dans la liste et en cliquant sur le bouton Désactiver.

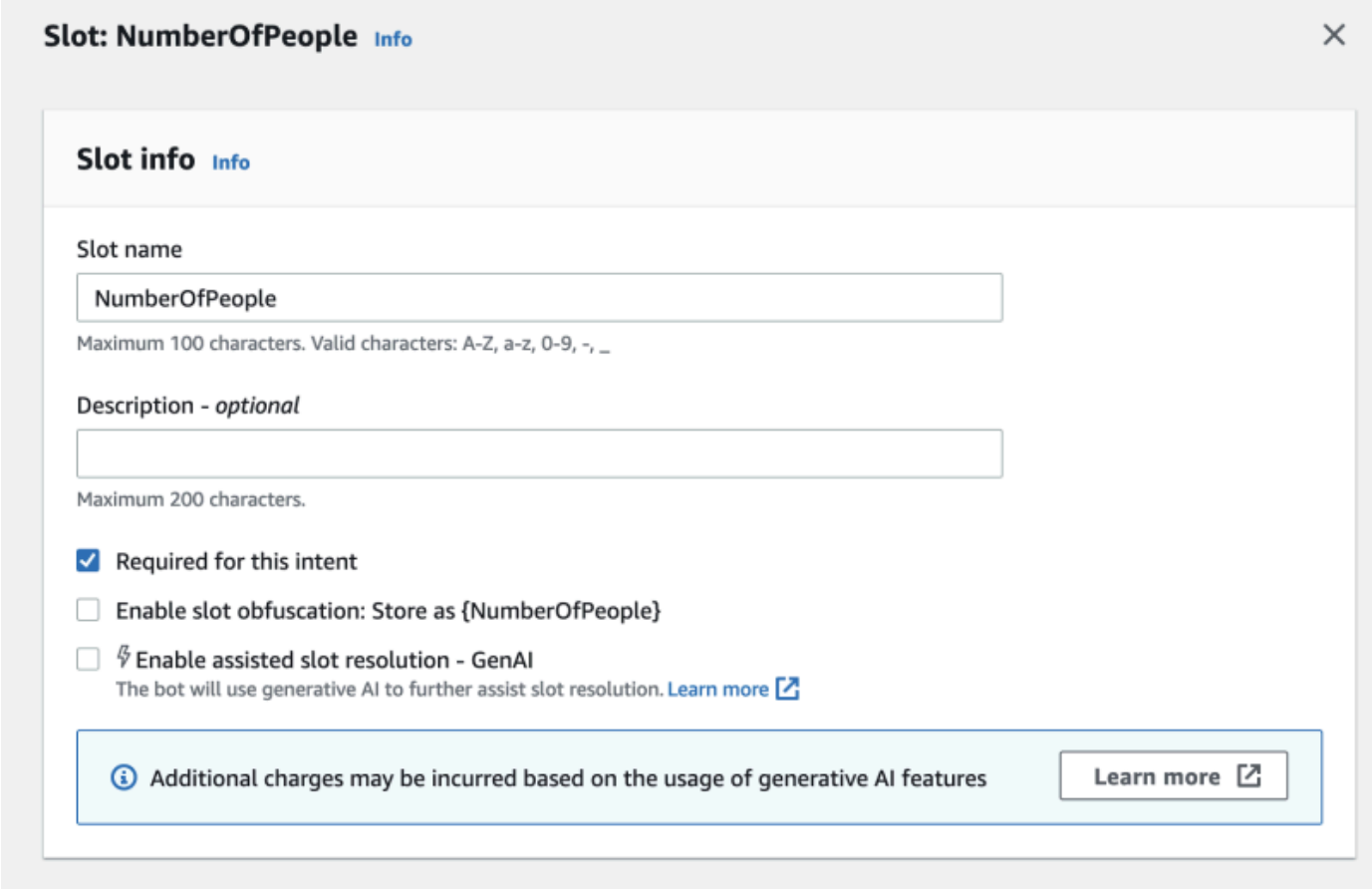
Activez la résolution assistée des emplacements dans les paramètres des emplacements

Vous pouvez activer la résolution assistée pour les emplacements intégrés pris en charge en accédant au niveau des emplacements pour chaque intention comportant des emplacements. Les emplacements doivent être l'un des emplacements intégrés pris en charge répertoriés ci-dessus pour avoir la possibilité d'activer la résolution assistée des emplacements. Si l'emplacement ne dispose pas de l'option permettant d'activer la résolution assistée des emplacements, l'option sera grisée.

> **Note**
>
> Vous devez d'abord activer la fonction de résolution assistée des emplacements sur le panneau Generative AI afin d'activer la fonction pour les emplacements individuels.

1. Connectez-vous à l'AWS Management Console et ouvrez la console Amazon Lex V2 à l'adresse https://console.aws.amazon.com/lexv2/home.

2. Dans le volet de navigation, sous Bots, sélectionnez le bot que vous souhaitez utiliser pour la résolution assistée des créneaux.
3. Sous Toutes les langues, sélectionnez Anglais (États-Unis) pour développer la liste.
4. Dans le panneau de gauche, choisissez Intentions pour afficher la liste des intentions du bot que vous avez sélectionné.
5. Dans l'écran Intentions, choisissez l'intention qui contient les emplacements que vous souhaitez modifier.
6. Sélectionnez le nom de l'intention pour afficher les emplacements correspondant à cette intention.
7. Sélectionnez le bouton Options avancées dans la section Machines à sous.
8. Cochez la case Activer la résolution assistée des emplacements pour activer la fonctionnalité.



9. Cliquez sur le bouton Update Slot dans le coin inférieur droit de l'écran. Cela activera la résolution assistée des créneaux pour les créneaux que vous avez choisis.

Vous pouvez activer la résolution assistée des emplacements intégrés pris en charge en effectuant des appels d'API.

- Suivez les étapes décrites Optimisez la création et les performances des robots grâce à l'IA générative pour activer la résolution assistée des emplacements pour les paramètres régionaux de votre bot.
- Envoyez une UpdateSlotdemande en spécifiant le créneau pour lequel vous souhaitez activer la résolution assistée des créneaux. Dans le `slotResolutionSetting` champ, définissez la `slotResolutionStrategy` valeur comme`EnhancedFallback`. Pour créer un nouvel emplacement avec la résolution d'intervalle assistée activée, envoyez plutôt une CreateSlotdemande.

Autorisations pour la résolution assistée des créneaux

- Pour accéder à cette fonctionnalité sur la console Amazon Lex V2, assurez-vous que votre rôle de console est `bedrock:ListFoundationModels` autorisé.
- Le rôle IAM associé au bot doit être `bedrock:InvokeModel` autorisé. Lorsque vous activez cette fonctionnalité avec la console Amazon Lex, la politique est automatiquement ajoutée au rôle de bot, à condition que votre bot utilise un rôle lié à un service généré par Amazon Lex.

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [
              "bedrock:InvokeModel"
          ],
          "Resource": [
              "arn:aws:bedrock:Region::foundation-model/modelId"
          ]
      }
  ]
}
```

Intention d'Amazon.QNA

> **Note**
>
> Avant de pouvoir tirer parti des fonctionnalités de l'IA générative, vous devez remplir les conditions préalables suivantes :
>
> 1. Accédez à la console Amazon Bedrock et inscrivez-vous pour accéder au modèle Anthropic Claude que vous souhaitez utiliser (pour plus d'informations, voir Accès au modèle). Pour plus d'informations sur les tarifs d'utilisation d'Amazon Bedrock, consultez les tarifs d'Amazon Bedrock.
> 2. Activez les fonctionnalités d'IA générative pour les paramètres régionaux de votre bot. Pour ce faire, suivez les étapes indiquées surOptimisez la création et les performances des robots grâce à l'IA générative.

Vous pouvez tirer parti d'Amazon Bedrock FM pour répondre aux questions des clients dans le cadre d'une conversation avec un bot. Amazon Lex V2 propose une fonction intégrée `AMAZON.QnAIntent` que vous pouvez ajouter à votre bot. Cette intention exploite les capacités d'intelligence artificielle générative d'Amazon Bedrock en identifiant les questions des clients et en recherchant une réponse dans les magasins de connaissances suivants (par exemple,**`Can you provide me details on the baggage limits for my international flight?`**). Cette fonctionnalité réduit le besoin de configurer les questions et réponses à l'aide d'un dialogue axé sur les tâches dans le cadre d'Amazon Lex V2. Cette intention reconnaît également les questions de suivi (par exemple,**`What about domestic flight?`**) sur la base de l'historique des conversations et fournit la réponse en conséquence.

Assurez-vous que votre rôle IAM dispose des autorisations appropriées pour y accéder `AMAZON.QnAIntent` en suivant les étapes décrites dansAutorisations pour le AMAZON.QnAIntent.

Pour en bénéficier, `AMAZON.QnAIntent` vous devez avoir configuré l'une des banques de connaissances suivantes.

- Base OpenSearch de données Amazon Service — Pour plus d'informations, consultez Création et gestion de domaines Amazon OpenSearch Service.
- Index Amazon Kendra — Pour plus d'informations, consultez Création d'un index.
- Base de connaissances Amazon Bedrock — Pour plus d'informations, consultez Création d'une base de connaissances.

Vous pouvez configurer `AMAZON.QnAIntent` le de deux manières :

Pour configurer à l'aide de configurations d'IA générative

1. Dans la console Amazon Lex V2, sélectionnez Bots dans le volet de navigation de gauche et choisissez le bot pour lequel vous souhaitez ajouter l'intention dans la section Bots.
2. Dans le volet de navigation de gauche, sélectionnez la langue pour laquelle vous souhaitez ajouter l'intention.
3. Dans la section Configurations génératives de l'IA, sélectionnez Configurer.
4. Dans la section Configurations QnA, sélectionnez Créer une intention QnA.

À configurer en ajoutant une intention intégrée à votre bot

1. Dans la console Amazon Lex V2, sélectionnez Bots dans le volet de navigation de gauche et choisissez le bot pour lequel vous souhaitez ajouter l'intention dans la section Bots.
2. Dans le volet de navigation de gauche, sélectionnez Intentions dans la langue pour laquelle vous souhaitez ajouter l'intention.
3. Sélectionnez Ajouter une intention, puis choisissez Utiliser une intention intégrée dans le menu déroulant.
4. Pour plus de détails sur les configurations du`AMAZON.QnAIntent`, consultezAMAZON.QnAIntent.

> **Note**
>
> Le `AMAZON.QnAIntent` est activé lorsqu'un énoncé n'est classé dans aucune des autres intentions présentes dans le bot. Cette intention est activée lorsqu'un énoncé n'est classé dans aucune des autres intentions présentes dans le bot. Notez que cette intention ne sera pas activée pour les énoncés manqués lors de l'obtention d'une valeur d'intervalle. Une fois reconnu, il `AMAZON.QnAIntent` utilise le modèle Amazon Bedrock spécifié pour effectuer des recherches dans la base de connaissances configurée et répondre à la question du client.

Rubriques

- Autorisations pour le AMAZON.QnAIntent

Autorisations pour le AMAZON.QnAIntent

Pour accéder à cette fonctionnalité sur la console Amazon Lex V2, assurez-vous que votre rôle de console dispose `bedrock:ListFoundationModels` d'autorisations.

Le rôle IAM associé au bot doit disposer des autorisations suivantes requises pour`AMAZON.QnAIntent`. Le rôle de bot doit être autorisé à appeler`bedrock:InvokeModel`. Vous devez également joindre une déclaration pour chaque magasin de données que vous spécifiez dans celui de vos robots `AMAZON.QnAIntent` (voir les `Permissions to access knowledge base in Amazon Bedrock` instructions `Permissions to access Amazon Kendra indexPermissions to access OpenSearch Service index`, et dans la politique ci-dessous). Lorsque vous activez cette fonctionnalité avec la console Amazon Lex, les politiques sont automatiquement ajoutées au rôle de bot, à condition que votre bot utilise un rôle lié à un service généré par Amazon Lex.

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
          "Sid": "Permissions to invoke Amazon Bedrock foundation models",
          "Effect": "Allow",
          "Action": [
              "bedrock:InvokeModel"
          ],
          "Resource": [
              "arn:aws:bedrock:region::foundation-model/model-id"
          ]
      },
      {
        "Sid": "Permissions to access Amazon Kendra index",
        "Effect": "Allow",
        "Action": [
            "kendra:Query",
            "kendra:Retrieve"
        ],
        "Resource": [
            "arn:aws:kendra:region:account-id:index/kendra-index"
        ]
     },
     {
        "Sid": "Permissions to access OpenSearch Service index",
        "Effect": "Allow",
```

```
        "Action": [
            "es:ESHttpGet",
            "es:ESHttpPost"
        ],
        "Resource": [
            "arn:aws:es:region:account-id:domain/domain-name/index-name/_search"
        ]
    },
    {
        "Sid": "Permissions to access knowledge base in Amazon Bedrock",
        "Effect": "Allow",
        "Action": [
            "bedrock:Retrieve"
        ],
        "Resource": [
            "arn:aws:bedrock:region:account-id:knowledge-base/knowledge-base"
        ]
    }
  ]
}
```

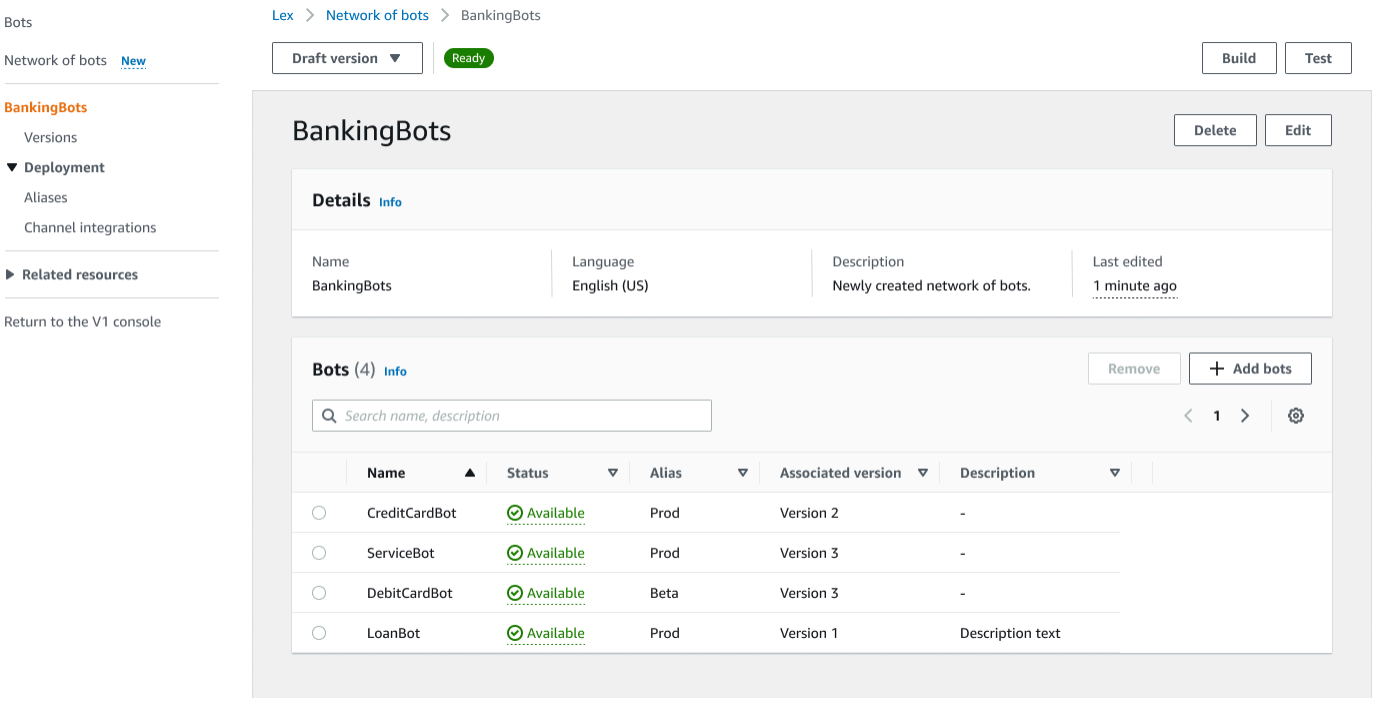
Création d'un réseau de robots

Network of Bots permet aux entreprises de proposer une expérience utilisateur unifiée via plusieurs robots. Avec Network of Bots, les entreprises peuvent ajouter plusieurs robots à un seul réseau afin de permettre une gestion flexible et indépendante du cycle de vie des robots. Le réseau expose une interface unifiée unique à l'utilisateur final et achemine la demande vers le bot approprié en fonction des entrées de l'utilisateur.

Les équipes peuvent collaborer pour créer un réseau de robots répondant aux différents besoins de l'entreprise en gérant et en ajoutant des robots au réseau au fur et à mesure que des robots améliorés sont déployés en production. Les développeurs peuvent simplifier et accélérer le déploiement et les améliorations en intégrant plusieurs robots dans un seul réseau.

Network of bots n'est actuellement disponible que dans la langue en-US.

> **Note**
>
> Actuellement, un réseau de bots est limité à un seul compte. Vous ne pouvez pas ajouter de robots provenant d'autres comptes.



Créez un réseau de robots

Connectez-vous à la console Amazon Lex V2 AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/lexv2/home. Choisissez Réseau de robots dans le menu latéral. Vous devez avoir créé au moins un bot pour créer un réseau de robots.

Étape 1 : Configuration des paramètres du réseau de robots

1. Dans la section Détails, entrez le nom de votre réseau et donnez-lui une description facultative.
2. Dans la section Autorisations IAM, choisissez un rôle AWS Identity and Access Management (IAM) qui fournit à Amazon Lex V2 l'autorisation d'accéder à d'autres AWS services, tels qu'Amazon. CloudWatch Vous pouvez demander à Amazon Lex V2 de créer le rôle, ou vous pouvez choisir un rôle existant avec CloudWatch des autorisations. Consultez la section Gestion des identités et des accès pour Amazon Lex V2 pour plus d'informations.
3. Dans la section Loi sur la protection de la vie privée des enfants en ligne (COPPA), choisissez la réponse appropriée. DataPrivacyPour plus d'informations, voir.
4. Dans la section Délai d'inactivité de la session, choisissez la durée pendant laquelle Amazon Lex V2 maintient une session ouverte avec un utilisateur. Amazon Lex V2 gère les variables de session pendant toute la durée de la session afin que votre bot puisse reprendre une conversation avec les mêmes variables. Reportez-vous à la section Définition du délai d'expiration de la session pour plus d'informations.
5. Dans la section Ajouter des paramètres de langue, choisissez une voix pour que votre bot interagisse avec les utilisateurs. Vous pouvez saisir une phrase dans l'extrait vocal et sélectionner Play pour écouter la voix.
6. Dans la section Paramètres avancés, ajoutez éventuellement des balises permettant d'identifier le bot. Les balises peuvent être utilisées pour contrôler l'accès et surveiller les ressources. Pour plus d'informations, consultez la section Ressources de balisage.
7. Choisissez Suivant pour créer le réseau de robots et passer à l'ajout de robots.

Étape 2 : Ajouter des robots

1. Dans la section Bots, sélectionnez + Ajouter des robots.
2. Une fenêtre modale Ajouter des robots s'affichera. Choisissez le bot à ajouter dans le menu déroulant Bot et l'alias du bot que vous souhaitez utiliser dans le menu déroulant Alias.

L'alias doit pointer vers une version numérotée du bot et non vers la version préliminaire. Vous pouvez ajouter jusqu'à 5 robots. Un bot peut être ajouté à un maximum de 25 réseaux différents.

3. Sélectionnez + Ajouter un bot pour ajouter d'autres robots à votre réseau. Pour supprimer un robot, sélectionnez Supprimer à côté du robot que vous souhaitez supprimer. Lorsque vous avez terminé d'ajouter des robots, choisissez Enregistrer pour fermer le modal.
4. Sélectionnez Enregistrer pour terminer la création de votre réseau.

Gérez votre réseau de robots

Après avoir créé votre réseau de robots, vous serez redirigé vers une page où vous pourrez gérer et développer votre réseau. Vous pouvez également accéder à cette page en sélectionnant Réseau de robots dans le menu latéral et en choisissant le nom du réseau à gérer.

1. Pour modifier les informations relatives à votre réseau, sélectionnez Modifier au-dessus de la section Détails. Pour supprimer le réseau, sélectionnez Supprimer au-dessus de la section Détails.
2. Dans la section Bots, vous pouvez ajouter d'autres robots en sélectionnant + Ajouter des robots. Vous pouvez également ajouter des robots en accédant à la page Bots dans le menu latéral de la console Amazon Lex V2. Cliquez sur le bouton radio à côté du robot que vous souhaitez ajouter, puis sélectionnez Ajouter à un réseau de robots dans le menu déroulant Actions.

   Dans le menu déroulant Réseau de robots du modal qui s'affiche, choisissez le réseau auquel vous souhaitez ajouter le bot. Choisissez ensuite l'alias du bot que vous souhaitez utiliser dans le menu déroulant Alias du bot. Sélectionnez Ajouter pour ajouter le bot au réseau que vous avez choisi.
3. Vous pouvez supprimer des robots de votre réseau en appuyant sur le bouton radio situé à côté d'un robot et en choisissant Supprimer.
4. Lorsque vous avez terminé de configurer votre réseau, sélectionnez Créer en haut à droite pour créer votre réseau. La création peut prendre quelques minutes. Si la compilation est réussie, une bannière verte de réussite apparaît en haut de la page.
5. Une fois le réseau créé, vous pouvez sélectionner Test en haut à droite pour qu'une fenêtre de discussion apparaisse dans le coin inférieur droit. Vous pouvez utiliser cette fenêtre de discussion pour discuter avec les robots de votre réseau et vous assurer que les flux de conversation et les transitions sont correctement configurés.

> **Note**
>
> Si vous ajoutez, supprimez ou mettez à jour des robots au sein de votre réseau, vous devez reconstruire le réseau.

Versions

Vous pouvez créer différentes versions de votre réseau de robots. Pour gérer les versions, choisissez votre réseau dans le menu latéral de la console Amazon Lex V2 et sélectionnez Versions.

1. Sélectionnez Créer une version pour créer une nouvelle version de votre réseau de robots. Vous pouvez ajouter une description facultative. Choisissez Créer pour créer la version.
2. Lorsque vous activez le bouton radio situé à côté d'une version de votre réseau de robots, vous pouvez sélectionner Associer un alias à la version pour pointer un alias vers cette version.
3. Pour gérer une version de votre réseau, sélectionnez le nom de la version dans la section Versions. Sur la page suivante, vous pouvez modifier les détails de la version et gérer les robots au sein de la version et de l'alias associé.

Alias

Vous pouvez utiliser des alias pour déployer vos réseaux. Pour gérer les alias, choisissez votre réseau dans le menu latéral de la console Amazon Lex V2 et sélectionnez Alias.

1. Sélectionnez Créer un alias pour créer un nouvel alias.
2. Donnez un nom à l'alias et une description facultative dans la section Détails de l'alias. Vous pouvez choisir une version à associer à l'alias dans la section Associer à une version et ajouter des balises dans la section Balises. Choisissez Créer pour créer l'alias.
3. Pour gérer un alias pour votre réseau, sélectionnez le nom de l'alias dans la section Alias. Sur la page suivante, vous pouvez modifier les détails de l'alias et gérer ses balises, ses intégrations de canaux et sa politique basée sur les ressources. Vous pouvez également consulter l'historique de son association avec les versions du réseau.

Intégrations de canaux

Pour intégrer votre réseau de robots à une plateforme de messagerie, choisissez votre réseau de robots dans le menu latéral de la console Amazon Lex V2. Sélectionnez ensuite Intégrations de canaux.

1. Sélectionnez Ajouter une chaîne pour intégrer votre réseau à une nouvelle chaîne.
2. Dans la section Plateforme, choisissez la plateforme sur laquelle vous souhaitez déployer votre bot dans Select platform. Un rôle IAM sera créé. Choisissez une clé dans le menu déroulant situé sous Clé KMS pour protéger vos informations.
3. Dans le canal de configuration de l'intégration, entrez le nom et une description facultative. Choisissez un alias dans le menu déroulant.
4. Obtenez le SID de votre compte et votre jeton d'authentification sur la plateforme et remplissez les champs SID du compte et jeton d'authentification. Consultez la section Intégration de vos robots pour plus d'informations.
5. Sélectionnez Créer pour terminer l'intégration du canal.

> **Note**
>
> Le réseau de robots n'est actuellement pas disponible dans Amazon Connect Voice ou Chat.

Déploiement de bots

Après avoir créé et testé votre bot, celui-ci est prêt à être déployé pour interagir avec vos clients. Dans cette section, découvrez comment créer des versions de votre bot lorsque vous avez effectué une mise à jour. Utilisez des alias pour pointer vers les différentes versions de votre bot lorsqu'elles sont prêtes à être déployées. Découvrez comment intégrer vos robots aux plateformes de messagerie, aux applications mobiles et aux sites Web.

Rubriques

Gestion des versions et alias

Amazon Lex V2 prend en charge la création de versions et d'alias de robots et de réseaux de robots afin que vous puissiez contrôler l'implémentation utilisée par vos applications clientes. Une version agit comme un instantané numéroté de votre travail. Vous pouvez pointer un alias vers la version de votre bot que vous souhaitez mettre à la disposition de vos clients. Entre deux versions, vous pouvez continuer à mettre à jour la `Draft` version de votre bot sans affecter l'expérience utilisateur.

Versions

Amazon Lex V2 prend en charge la création de versions de bots afin que vous puissiez contrôler l'implémentation utilisée par vos applications clientes. Une version est un instantané numéroté de votre travail que vous pouvez créer pour être utilisé dans différentes parties de votre flux de travail, telles que le développement, le déploiement de la version bêta et la production.

La version préliminaire

Lorsque vous créez un bot Amazon Lex V2, il n'existe qu'une seule version, la `Draft` version.

`Draft`est la copie de travail de votre bot. Vous ne pouvez mettre à jour que la `Draft` version et, jusqu'à ce que vous créiez votre première version, `Draft` c'est la seule version du bot dont vous disposez.

La `Draft` version de votre bot est associée au`TestBotAlias`. `TestBotAlias`Il ne doit être utilisé que pour des tests manuels. Amazon Lex V2 limite le nombre de demandes d'exécution que vous pouvez envoyer à l'`TestBotAlias`alias du bot.

Création d'une version

Lorsque vous créez une version d'un bot Amazon Lex V2, vous créez un instantané numéroté du bot afin de pouvoir utiliser le bot tel qu'il existait au moment de la création de la version. Une fois que vous avez créé une version numérique, elle restera la même pendant que vous continuez à travailler sur la version préliminaire de votre application.

Lorsque vous créez une version, vous pouvez choisir les paramètres régionaux à inclure dans la version. Il n'est pas nécessaire de choisir tous les paramètres régionaux d'un bot. De plus, lorsque vous créez une version, vous pouvez choisir un paramètre régional à partir d'une version précédente. Par exemple, si vous disposez de trois versions d'un bot, vous pouvez choisir un paramètre régional dans la `Draft` version et un dans la version deux lorsque vous créez la version quatre.

Si vous supprimez un paramètre régional de la `Draft` version, il n'est pas supprimé d'une version numérotée.

Si la version d'un bot n'est pas utilisée pendant six mois, Amazon Lex V2 la marquera comme inactive. Lorsqu'une version est inactive, vous ne pouvez pas utiliser les opérations d'exécution avec le bot. Pour activer le bot, reconstruisez toutes les langues associées à la version.

Mettre à jour un bot Amazon Lex V2

Vous ne pouvez mettre à jour que la `Draft` version d'un bot Amazon Lex V2. Les versions ne peuvent pas être modifiées. Vous pouvez créer une nouvelle version à tout moment après avoir mis à jour une ressource dans la console ou au cours de l'CreateBotVersionopération.

Supprimer un bot ou une version d'Amazon Lex V2

Amazon Lex V2 prend en charge la suppression d'un bot ou d'une version à l'aide de la console ou de l'une des opérations de l'API :

- DeleteBot
- DeleteBotVersion

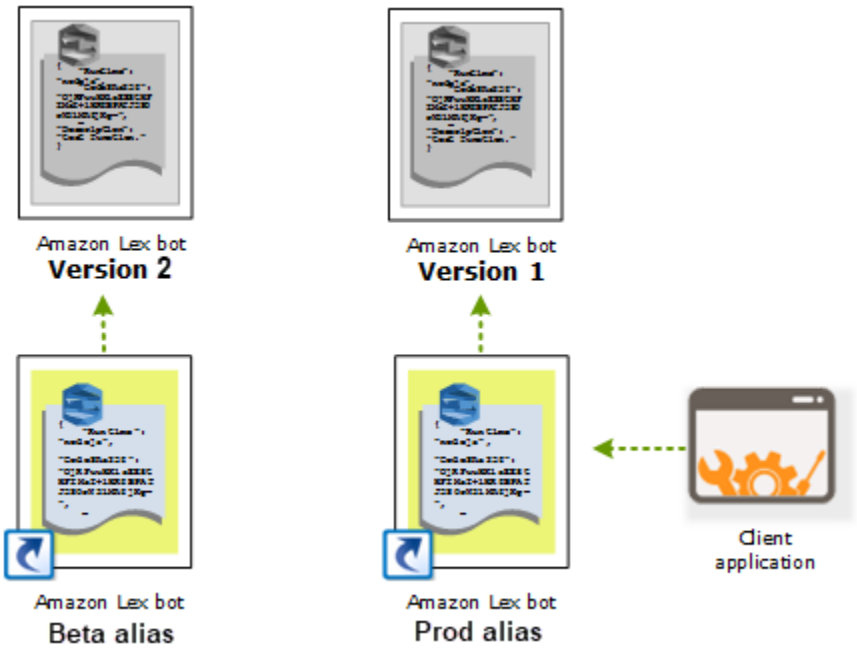
Alias

Les robots Amazon Lex V2 prennent en charge les alias. Un alias est un pointeur vers une version spécifique d'un bot. Avec un alias, vous pouvez facilement mettre à jour la version que vos applications clientes utilisent. Par exemple, vous pouvez faire pointer un alias vers la version 1 de votre bot. Lorsque vous êtes prêt à mettre à jour le bot, vous créez la version 2 et modifiez l'alias pour qu'il pointe vers la nouvelle version. Comme vos applications utilisent l'alias au lieu d'une version spécifique, tous vos clients obtiennent les nouvelles fonctionnalités sans avoir besoin d'être mis à jour.

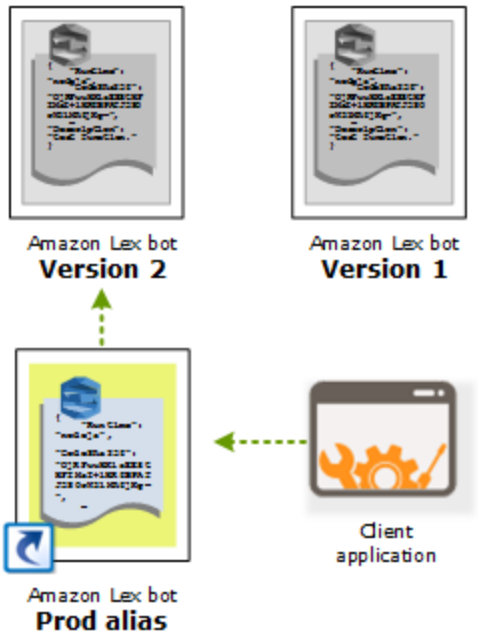
Un alias est un pointeur vers une version spécifique d'un bot Amazon Lex V2. Utilisez un alias pour permettre aux applications clientes d'utiliser une version du bot sans que les applications aient besoin de savoir de quelle version spécifique il s'agit.

Lorsque vous créez un bot, Amazon Lex V2 crée un alias appelé `TestBotAlias` que vous pouvez utiliser pour tester votre bot. L'`TestBotAlias`alias est toujours associé à la `Draft` version de votre bot. Vous ne devez utiliser l'`TestBotAlias`alias qu'à des fins de test. Amazon Lex V2 limite le nombre de demandes d'exécution que vous pouvez envoyer à l'alias.

L'exemple suivant montre deux versions d'un bot Amazon Lex V2, la version 1 et la version 2. Chacune de ces versions de bot est associée à un alias, BETA et PROD, respectivement. Les applications clientes utilisent l'alias PROD pour accéder au bot.



Lorsque vous créez une deuxième version du bot, vous pouvez mettre à jour l'alias pour qu'il renvoie vers cette nouvelle version à l'aide de la console ou de l'opération UpdateBotAlias. Lorsque vous modifiez l'alias, toutes vos applications clientes utilisent la nouvelle version. S'il existe un problème lié à la nouvelle version, vous pouvez restaurer la version précédente en changeant simplement l'alias pour qu'il pointe vers cette version.



Lorsque vous configurez vos applications clientes pour appeler les API Amazon Lex Runtime V2 afin de permettre aux clients d'interagir avec votre bot, vous utilisez l'alias qui indique la version que vous souhaitez que vos clients utilisent.

> **Note**
>
> Bien que vous puissiez tester la `Draft` version d'un bot dans la console, nous vous recommandons, lorsque vous intégrez un bot à votre application cliente, de créer d'abord une version et de créer un alias pointant vers cette version. Utilisez l'alias dans votre application cliente pour les raisons expliquées dans cette section. Lorsque vous mettez à jour un alias, Amazon Lex V2 utilise la version actuelle pour toutes les sessions en cours. Les nouvelles sessions utilisent la nouvelle version.

Utilisation d'une application Java pour interagir avec un bot Amazon Lex V2

La AWS SDK for Javaversion 2.0 fournit une interface que vous pouvez utiliser à partir de vos applications Java pour interagir avec vos robots. Utilisez le SDK for Java client pour créer des applications client pour les utilisateurs.

L'application suivante interagit avec le OrderFlowers bot que vous avez créé dansExercice 1 : créer un bot à partir d'un exemple. Il utilise le`LexRuntimeV2Client` fichier depuis le SDK for Java pour appeler l'RecognizeTextopération afin de mener une conversation avec le bot.

La sortie de la conversation ressemble à suit suit suit suit suit suit suit suit suit suit suit suit

```
User : I would like to order flowers
Bot  : What type of flowers would you like to order?
User : 1 dozen roses
Bot  : What day do you want the dozen roses to be picked up?
User : Next Monday
Bot  : At what time do you want the dozen roses to be picked up?
User : 5 in the evening
Bot  : Okay, your dozen roses will be ready for pickup by 17:00 on 2021-01-04.  Does
 this sound okay?
User : Yes
Bot  : Thanks.
```

Pour les structures JSON qui sont envoyées entre l'application cliente et le bot Amazon Lex V2, consultezExercice 2 : Passez en revue le flux de conversation.

Pour exécuter l'application, vous devez renseigner les informations suivantes.

- BotID comme suit suit suit suit suit suit suit suit suit suit suit suit suit suit suit suit suit suit suit suit suit suit suit suit suit suit suit suit suit suit Vous pouvez voir l'ID du bot dans la console Amazon Lex V2 sur la page des paramètres du bot.
- botAliasId — L'ID se présente comme suit suit suit suit suit suit suit suit suit suit suit suit suit suit suit suit suit suit suit suit suit suit suit suit suit suit suit suit suit suit Vous pouvez consulter l'identifiant d'alias du bot dans la console Amazon Lex V2 sur la page Alias. Si vous ne voyez pas l'identifiant d'alias dans la liste, cliquez sur l'icône en forme d'engrenage en haut à droite et activez Alias ID.

- localeID — L'identifiant des paramètres régionaux que vous avez utilisés pour votre bot. Pour obtenir la liste des paramètres régionaux, reportez-vous à la sectionLangues et paramètres régionaux pris en charge par Amazon Lex V2.
- accessKey et secretKey : clés d'authentification de votre compte. Si vous n'en avez pas, créez-en un à l'aide de laAWS Identity and Access Management console.
- SessionId : identifiant de la session avec le bot Amazon Lex V2. Dans ce cas, le code utilise un UUID aléatoire.
- région USA Est (Virginie du Nord), assurez-vous de changer la région USA Est (Virginie du Nord), assurez-vous de changer la région USA Est (Virginie du Nord).

Les applications utilisent une fonction appeléegetRecognizeTextRequest à créer des demandes individuelles au bot. La fonction crée une requête avec les paramètres requis à envoyer à Amazon Lex V2.

```
package com.lex.recognizetext.sample;

import software.amazon.awssdk.auth.credentials.AwsBasicCredentials;
import software.amazon.awssdk.auth.credentials.AwsCredentialsProvider;
import software.amazon.awssdk.auth.credentials.StaticCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lexruntimev2.LexRuntimeV2Client;
import software.amazon.awssdk.services.lexruntimev2.model.RecognizeTextRequest;
import software.amazon.awssdk.services.lexruntimev2.model.RecognizeTextResponse;

import java.net.URISyntaxException;
import java.util.UUID;


/**
 * This is a sample application to interact with a bot using RecognizeText API.
 */
public class OrderFlowersSampleApplication {

    public static void main(String[] args) throws URISyntaxException,
 InterruptedException {
        String botId = "";
        String botAliasId = "";
        String localeId = "en_US";
        String accessKey = "";
```

```
        String secretKey = "";
        String sessionId = UUID.randomUUID().toString();
        Region region = Region.US_EAST_1; // pick an appropriate region

        AwsBasicCredentials awsCreds = AwsBasicCredentials.create(accessKey,
secretKey);
        AwsCredentialsProvider awsCredentialsProvider =
StaticCredentialsProvider.create(awsCreds);

        LexRuntimeV2Client lexV2Client = LexRuntimeV2Client
                .builder()
                .credentialsProvider(awsCredentialsProvider)
                .region(region)
                .build();

        // utterance 1
        String userInput = "I would like to order flowers";
        RecognizeTextRequest recognizeTextRequest = getRecognizeTextRequest(botId,
botAliasId, localeId, sessionId, userInput);
        RecognizeTextResponse recognizeTextResponse =
lexV2Client.recognizeText(recognizeTextRequest);

        System.out.println("User : " + userInput);
        recognizeTextResponse.messages().forEach(message -> {
            System.out.println("Bot : " + message.content());
        });

        // utterance 2
        userInput = "1 dozen roses";
        recognizeTextRequest = getRecognizeTextRequest(botId, botAliasId, localeId,
sessionId, userInput);
        recognizeTextResponse = lexV2Client.recognizeText(recognizeTextRequest);

        System.out.println("User : " + userInput);
        recognizeTextResponse.messages().forEach(message -> {
            System.out.println("Bot : " + message.content());
        });

        // utterance 3
        userInput = "next monday";
        recognizeTextRequest = getRecognizeTextRequest(botId, botAliasId, localeId,
sessionId, userInput);
        recognizeTextResponse = lexV2Client.recognizeText(recognizeTextRequest);
```

```
        System.out.println("User : " + userInput);
        recognizeTextResponse.messages().forEach(message -> {
            System.out.println("Bot : " + message.content());
        });

        // utterance 4
        userInput = "5 in evening";
        recognizeTextRequest = getRecognizeTextRequest(botId, botAliasId, localeId,
 sessionId, userInput);
        recognizeTextResponse = lexV2Client.recognizeText(recognizeTextRequest);

        System.out.println("User : " + userInput);
        recognizeTextResponse.messages().forEach(message -> {
            System.out.println("Bot : " + message.content());
        });

        // utterance 5
        userInput = "Yes";
        recognizeTextRequest = getRecognizeTextRequest(botId, botAliasId, localeId,
 sessionId, userInput);
        recognizeTextResponse = lexV2Client.recognizeText(recognizeTextRequest);

        System.out.println("User : " + userInput);
        recognizeTextResponse.messages().forEach(message -> {
            System.out.println("Bot : " + message.content());
        });
    }

    private static RecognizeTextRequest getRecognizeTextRequest(String botId, String
 botAliasId, String localeId, String sessionId, String userInput) {
        RecognizeTextRequest recognizeTextRequest = RecognizeTextRequest.builder()
                .botAliasId(botAliasId)
                .botId(botId)
                .localeId(localeId)
                .sessionId(sessionId)
                .text(userInput)
                .build();
        return recognizeTextRequest;
    }
}
```

Résilience globale

> **Note**
>
> Cette fonctionnalité est disponible uniquement pour les instances Amazon Connect et Amazon Lex V2 créées dans les régions USA Est (Virginie du Nord) et USA Ouest (Oregon). Pour accéder à cette fonctionnalité, contactez votre architecte de solutions Amazon Connect ou votre responsable de compte technique.

Global Resiliency vous permet de répliquer un bot dans une région secondaire. La région secondaire peut être activée grâce à la réplication automatique du bot de l'utilisateur dans les deux régions. Vous disposerez d'une région de secours en cas de panne régionale. Une fois que Global Resiliency est actif, les nouveaux bots créés sont répliqués dans une deuxième AWS région.

Après avoir activé cette fonctionnalité, vous pouvez automatiser la réplication des robots Amazon Lex V2 ainsi que de leurs ressources, versions et alias dans une AWS région associée en temps quasi réel. Grâce à cette fonctionnalité, vous pouvez surveiller le numéro de version du bot d'origine et de la réplique pour vous assurer que la réplique du bot reste synchronisée avec le bot d'origine. Lorsque vous activez la réplication, vous pouvez activer la AWS région prédéfinie dans laquelle vous souhaitez que le bot soit répliqué (les régions sont basées sur des paires prédéterminées). Toute mise à jour du bot source dans la région source est automatiquement mise à jour sur le bot répliqué dans la deuxième région.

> **Note**
>
> Lorsque Global Resiliency est activé, seuls les bots, les versions et les alias créés après l'activation de la fonctionnalité seront répliqués dans la région répliquée. Les robots, versions et alias créés précédemment ne seront pas présents dans la région répliquée. La seconde région identifiée est en lecture seule et par paires prédéterminées. Les mises à jour du bot sont limitées à la région dans laquelle le bot a été initialement créé.

Informations supplémentaires sur l'utilisation de Global Resiliency :

- Global Resiliency ne fonctionne actuellement qu'avec des paires de régions prédéterminées.

us-east-1	us-west-2
eu-west-2	eu-central-1

- Vous pouvez créer une réplique de n'importe quel bot Amazon Lex V2. Vous devez créer une nouvelle version et un nouvel alias pour le bot une fois Global Resiliency activé.
- Les alias activés dans Global Resiliency ne peuvent être associés qu'aux versions compatibles avec Global Resiliency.

Limites:

- Global Resiliency ne reproduit pas les bots créés avec des emplacements utilisant le LLM, tels que CFAQ et Utterance Generation.
- Global Resiliency ne reproduit pas un réseau de robots, mais tout bot faisant partie du réseau de robots peut toujours être répliqué individuellement.

Rubriques

Autorisations pour répliquer des robots et gérer les répliques de robots

Si la AmazonLexFullAccesspolitique est attachée à un rôle IAM, il peut créer et gérer des répliques de robots.

Si vous préférez créer un rôle avec des autorisations minimales pour Global Resiliency, utilisez la politique suivante, qui contient les déclarations suivantes.

- Autorisations d'accès au rôle lié au service Amazon Lex V2 pour la réplication de robots.
- Autorisations permettant à Amazon Lex V2 de créer un rôle lié à un service pour la réplication de robots en votre nom.
- Autorisations permettant d'appeler les API de réplication du bot.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "GetReplicationSLR",
            "Effect": "Allow",
            "Action": [
                "iam:GetRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/replication.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Replication*"
            ]
        },
        {
            "Sid": "CreateReplicationSLR",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole",
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/replication.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Replication*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "lexv2.amazonaws.com"
                }
            }
        },
        {
            "Sid": "AllowBotReplicaActions",
            "Effect": "Allow",
            "Action": [
                "lex:CreateBotReplica",
                "lex:DescribeBotReplica",
                "lex:ListBotReplica",
                "lex:ListBotVersionReplicas",
                "lex:ListBotAliasReplicas",
                "lex:DeleteBotReplica"
            ],
            "Resource": [
                "arn:aws:lex:*:*:bot/*",
                "arn:aws:lex:*:*:bot-alias/*"
            ]
```

```
        }
    ]
}
```

Vous pouvez restreindre davantage les autorisations en les modifiant comme suit.

- Remplacez * par des identifiants de bot ou d'alias de bot spécifiques pour limiter les autorisations à des robots ou à des alias de bot spécifiques.
- Utilisez un sous-ensemble d'`lex BotReplica`actions pour limiter le rôle à des actions spécifiques.

Pour obtenir un exemple, consultez Autoriser les utilisateurs à créer et à afficher des répliques de robots, mais pas à les supprimer.

Déployer la résilience mondiale

Panneau d'information sur la résilience mondiale

Vous pouvez accéder aux informations suivantes dans le panneau Global Resiliency :

- Détails de la source : informations sur la région source de votre bot, le type de réplique, la date d'activation de la réplication et la dernière version créée. Utilisez ces informations pour suivre les itérations de votre bot.
- Détails de réplication : après avoir créé la réplique de votre bot, vous pouvez suivre la région répliquée, le type de réplique, la date de synchronisation de la dernière version et la dernière version répliquée. Utilisez ces informations pour suivre la synchronisation de la réplique de votre bot.
- Région source : région dans laquelle la résilience globale est activée. Vous pouvez apporter des modifications dans la région source pour répliquer le bot dans les deux régions.
- Type de réplique : indique si le bot est en lecture seule ou s'il est capable de lire et d'écrire en fonction de la région.
- Région de réplication : région secondaire utilisée pour répliquer votre bot source pour Global Resiliency. Global Resiliency ne fonctionne actuellement qu'avec les paires régionales IAD/PDX et LDN/FRA.
- Date d'activation de la réplication : date et heure d'activation de la réplique du bot.
- Dernière version créée : dernière version du bot associée à la réplique dans la région source.

Favoriser la résilience mondiale

> **Note**
>
> Cette fonctionnalité est disponible uniquement pour les instances Amazon Connect et Amazon Lex V2 créées dans les régions USA Est (Virginie du Nord) et USA Ouest (Oregon). Pour accéder à cette fonctionnalité, contactez votre architecte de solutions Amazon Connect ou votre responsable de compte technique.

Avant d'activer Global Resiliency dans la console Amazon Lex V2, vous devez vous assurer que l'utilisateur qui active la réplication des robots est autorisé à créer des rôles liés aux services (SLR). Global Resiliency utilisera ces informations d'identification FAS pour créer un SLR dans le compte activé lorsqu'il CreateReplica est invoqué. Pour plus d'informations sur la configuration du SLR pour une résilience globale dans Amazon Lex V2, consultez la politique gérée par AWS :. AmazonLexFullAccess

Activez Global Resiliency et configurez la réplication des robots pour une deuxième région :

1. Connectez-vous à la console de gestion AWS et ouvrez la console Amazon Lex à l'adresse https://console.aws.amazon.com/lex/.
2. Choisissez le bot que vous souhaitez répliquer dans la barre de navigation Bots sur le panneau de navigation de gauche.
3. Choisissez Déploiement > Résilience globale.
4. Cliquez sur le bouton Créer une réplique dans le coin supérieur droit de la fenêtre pour créer une version préliminaire de votre bot.

   > **Note**
   >
   > Vérifiez qu'aucun robot ne porte le même nom que le bot que vous souhaitez répliquer dans la région secondaire. (Votre bot doit porter un nom unique).

5. Accédez à Global Resiliency, cliquez sur Créer une réplique. Cette action crée une version préliminaire de votre bot. (vous n'avez pas besoin de retourner à l'onglet Global Resiliency, sauf pour consulter le statut ou les détails des futures versions).

> **Note**
>
> Vous pouvez également créer un bot Alias pour la réplication dans Global Resiliency en accédant à Alias et en sélectionnant Créer un nouvel alias pour le bot activé par Global Resiliency. Seuls les alias créés après l'activation de la réplication seront répliqués.

6. Accédez à Alias : créez un nouvel alias pour le bot activé par Global Resiliency. Seuls les alias créés après l'activation de la réplication seront répliqués.
7. Accéder à la version : créez une nouvelle version pour le bot activé par Global Resiliency. Seules les versions créées après l'activation de la réplication seront répliquées.

> **Note**
>
> Les clients ont toujours le contrôle total de la gestion de leurs politiques basées sur les ressources et des balises pour les robots répliqués. Les fonctions Lambda et CloudWatch les groupes de journaux devront être déployés dans les deux régions avec les mêmes identifiants. Les utilisateurs n'auront pas à associer à nouveau la fonction lambda dans la région de réplication.

Désactiver la résilience globale

Vous pouvez désactiver la résilience globale à tout moment en sélectionnant le bouton Désactiver la résilience globale. Cette action empêche la réplication de votre bot source et de tous les alias et versions qui lui sont associés dans d'autres régions.

Utiliser les API avec une résilience globale

Vous pouvez effectuer des appels d'API dans Global Resiliency à l'aide des API suivantes. Des informations supplémentaires sur les API Global Resiliency et Amazon Lex V2 sont disponibles dans le guide des API Amazon Lex V2.

- CreateBotReplica

  Activez Global Resiliency et créez un bot répliqué. Nécessite `replicaRegion`.

  Pour plus d'informations, consultez CreateBotReplicale guide de l'API Lex.

- DeleteBotReplica

  Désactivez Global Resiliency et supprimez le bot répliqué. Nécessite `replicaRegion` et `botId`.

  Pour plus d'informations, consultez DeleteBotReplicale guide de l'API Lex.

- ListBotReplicas

  Répertoriez les robots répliqués dans la zone secondaire. Nécessite `botId`.

  Pour plus d'informations, consultez ListBotReplicasle guide de l'API Lex.

- DescribeBotReplica

  Résumé des informations relatives au bot répliqué. Nécessite `replicaRegion` et `botId`.

  Pour plus d'informations, consultez DescribeBotReplicale guide de l'API Lex.

Intégration d'un bot Amazon Lex V2 à une plateforme de messagerie

Cette section explique comment intégrer les robots Amazon Lex V2 aux plateformes de messagerie Facebook, Slack et Twilio. Si vous ne possédez pas encore de bot Amazon Lex V2, créez-en un. Dans cette rubrique, nous partons du principe que vous utilisez le bot que vous avez créé dansExercice 1 : créer un bot à partir d'un exemple. Cependant, vous pouvez utiliser n'importe quel bot.

> **Note**
>
> Lorsque vous stockez vos configurations Facebook, Slack ou Twilio, Amazon Lex V2 utilise un AWS KMS key pour chiffrer les informations. La première fois que vous créez un canal vers l'une de ces plateformes de messagerie, Amazon Lex V2 crée une clé gérée par le client par défaut (`aws/lex`) dans votre AWS compte ou vous pouvez sélectionner votre propre clé gérée par le client. Amazon Lex V2 ne prend en charge que les clés symétriques. Pour plus d'informations, consultez le Guide du développeur AWS Key Management Service.

Lorsqu'une plateforme de messagerie envoie une demande à Amazon Lex V2, elle inclut des informations spécifiques à la plate-forme en tant qu'attribut de demande pour votre fonction

Lambda. Utilisez cet attribut pour personnaliser le comportement de votre bot. Pour de plus amples informations, veuillez consulter Définition des attributs de demande.

Attribut de demande commun

Attribut	Description
x-amz-lex:canaux:plateforme	L'une des valeurs suivantes : • `Facebook` • `Slack` • `Twilio`

Intégration d'un bot Amazon Lex V2 à Facebook Messenger

Vous pouvez héberger votre robot Amazon Lex V2 sur Facebook Messenger. Lorsque vous le faites, les utilisateurs de Facebook peuvent interagir avec votre bot pour atteindre leurs objectifs.

Avant de commencer, vous devez créer un compte de développeur Facebook à l'adresse https://developers.facebook.com.

Procédez comme suit :

Rubriques

- Étape 1 : créer une application Facebook
- Étape 2 : Intégrer Facebook Messenger au bot Amazon Lex V2
- Étape 3 : intégration complète à Facebook
- Étape 4 : Tester l'intégration

Étape 1 : créer une application Facebook

Sur le portail des développeurs Facebook, créez une application et une page Facebook.

Pour créer une application Facebook

1. Ouvrez https://developers.facebook.com/apps
2. Sélectionnez Create App (Créer une application).

3. Sur la page Créer une application, choisissez Business, puis Suivant.
4. Pour les champs Nom de l'application complémentaire, e-mail de contact de l'application et Compte professionnel, faites les choix appropriés pour votre application. Choisissez Créer une application pour continuer.
5. Dans Ajouter des produits à votre application, choisissez Configurer dans la vignette Messenger.
6. Dans la section Jetons d'accès, choisissez Ajouter ou supprimer des pages.
7. Choisissez la page à utiliser avec votre application, puis choisissez Suivant.
8. Dans Qu'est-ce que l'application est autorisée à faire ?, conservez les valeurs par défaut, puis choisissez OK.
9. Dans la page de confirmation, sélectionnez OK.
10. Dans la section Jetons d'accès, choisissez Générer un jeton, puis copiez le jeton. Vous entrez ce jeton dans la console Amazon Lex V2.
11. Dans le menu de gauche, choisissez Paramètres, puis Basic.
12. Dans le champ Secret de l'application, choisissez Afficher, puis copiez le secret. Vous entrez ce jeton dans la console Amazon Lex V2.

Étape suivante

Étape 2 : Intégrer Facebook Messenger au bot Amazon Lex V2

Étape 2 : Intégrer Facebook Messenger au bot Amazon Lex V2

Au cours de cette étape, vous associez votre bot Amazon Lex V2 à Facebook.

1. Connectez-vous à l'AWS Management Consoleet ouvrez la console Amazon Lex à l'adresse https://console.aws.amazon.com/lex/.
2. Dans la liste des robots, choisissez le robot Amazon Lex V2 que vous avez créé.
3. Dans le menu de gauche, choisissez Intégrations de chaînes, puis choisissez Ajouter une chaîne.
4. Dans Créer une chaîne, procédez comme suit :
    a. Dans le champ Plateforme, choisissez Facebook.
    b. Dans Politiques d'identité, choisissez laAWS KMS clé pour protéger les informations de la chaîne. La clé par défaut est fournie par Amazon Lex V2.

c. Pour la configuration de l'intégration, attribuez un nom au canal et une description facultative. Choisissez l'alias qui pointe vers la version du bot à utiliser et choisissez la langue prise en charge par la chaîne.

d. Pour Configuration supplémentaire, entrez ce qui suit :

- Alias : chaîne identifiant l'application qui appelle Amazon Lex V2. Vous pouvez utiliser n'importe quelle chaîne. Enregistrez cette chaîne, saisissez-la dans la console de développement de Facebook.
- Jeton d'accès à la page : jeton d'accès à la page que vous avez copié depuis la console de développement Facebook.
- Clé secrète de l'application : clé secrète que vous avez copiée depuis la console de développement Facebook.

e. Sélectionnez Create (Créer).

f. Amazon Lex V2 affiche la liste des chaînes de votre bot. Dans la liste, choisissez la chaîne que vous venez de créer.

g. Dans URL de rappel, enregistrez l'URL de rappel. Vous entrez cette URL dans la console de développement de Facebook.

Étape suivante

Étape 3 : intégration complète à Facebook

Étape 3 : intégration complète à Facebook

Au cours de cette étape, utilisez la console de développement Facebook pour terminer l'intégration à Amazon Lex V2.

Pour terminer l'intégration de Facebook Messenger

1. Ouvrez https://developers.facebook.com/apps
2. Dans la liste des applications, choisissez l'application que vous souhaitez intégrer à Facebook Messenger.
3. Dans le menu de gauche, choisissez Messenger, puis Paramètres.
4. Dans la section Webhooks :

   a. Choisissez Ajouter une URL de rappel.

b. Dans Modifier l'URL de rappel, entrez ce qui suit :

- URL de rappel : entrez l'URL de rappel que vous avez enregistrée depuis la console Amazon Lex V2.
- Jeton de vérification : entrez l'alias que vous avez saisi dans la console Amazon Lex V2.

c. Choisissez Verify and Save.

d. Choisissez Ajouter des abonnements sous Webhooks à côté de votre page.

e. Dans la fenêtre qui apparaît, choisissez`messages` puis cliquez sur Enregistrer.

Étape suivante

Étape 4 : Tester l'intégration

Étape 4 : Tester l'intégration

Vous pouvez désormais démarrer une conversation depuis Facebook Messenger avec votre robot Amazon Lex V2.

Pour tester l'intégration entre Facebook Messenger et un bot Amazon Lex V2

1. Ouvrez la page Facebook que vous avez associée à votre bot à l'étape 1.
2. Dans la fenêtre Messenger, utilisez les énoncés de test fournis dansExercice 1 : créer un bot à partir d'un exemple.

Intégration d'un bot Amazon Lex V2 à Slack

Cette rubrique fournit des instructions pour intégrer un bot Amazon Lex V2 à l'application de messagerie Slack. Procédez comme suit :

Rubriques

- Étape 1 : S'inscrire à Slack et créer une équipe Slack
- Étape 2 : créer une application Slack
- Étape 3 : Intégrer l'application Slack au bot Amazon Lex V2
- Étape 4 : Intégration complète à Slack
- Étape 5 : Test de l'intégration

Étape 1 : S'inscrire à Slack et créer une équipe Slack

Ouvrez un compte Slack et créez une équipe Slack. Pour obtenir des instructions, consultez Using Slack. Dans la section suivante, vous allez créer une application Slack, que n'importe quelle équipe Slack peut installer.

Étape suivante

Étape 2 : créer une application Slack

Étape 2 : créer une application Slack

Dans cette section, vous effectuez les opérations suivantes :

1. Créez une application Slack dans la console d'API Slack.
2. Configurez l'application pour ajouter une messagerie interactive à votre bot.

À la fin de cette section, vous trouverez les informations d'identification de l'application (ID client, secret client et jeton de vérification). À l'étape suivante, vous utiliserez ces informations pour intégrer le bot à la console Amazon Lex V2.

Pour créer une application Slack

1. Connectez-vous à la console d'API Slack à l'adresse https://api.slack.com.
2. Créez une application

   Une fois que vous avez créé l'application, Slack affiche la page Basic Information pour l'application.
3. Configurez les fonctions de l'application comme suit :

   - Dans le menu de gauche, choisissez Interactivité et raccourcis.

     - Choisissez le bouton bascule pour activer les composants interactifs.
     - Dans la zone Request URL, indiquez une URL valide. Par exemple, vous pouvez utiliser **`https://slack.com`**.

       > **Note**
       >
       > Pour le moment, saisissez n'importe quelle URL valide pour obtenir le jeton de vérification dont vous aurez besoin dans l'étape suivante. Vous mettrez à jour

cette URL après avoir ajouté l'association de chaînes de robots dans la console Amazon Lex.

- Choisissez Save Changes (Enregistrer les modifications).

4. Dans la section Paramètres du menu de gauche, sélectionnez Basic Information. Enregistrez les identifiants d'application suivants :

   - ID client
   - Clé secrète du client
   - Jeton de vérification

Étape suivante

Étape 3 : Intégrer l'application Slack au bot Amazon Lex V2

Étape 3 : Intégrer l'application Slack au bot Amazon Lex V2

Dans cette section, intégrez l'application Slack que vous avez créée au bot Amazon Lex V2 que vous avez créé à l'aide des intégrations de canaux.

1. Connectez-vous à l'AWS Management Consoleet ouvrez la console Amazon Lex à l'adresse https://console.aws.amazon.com/lex/.
2. Dans la liste des robots, choisissez le robot Amazon Lex V2 que vous avez créé.
3. Dans le menu de gauche, choisissez Intégrations de chaînes, puis choisissez Ajouter une chaîne.
4. Dans Créer une chaîne, procédez comme suit :

   a. Pour Platform, choisissez Slack.

   b. Dans Politiques d'identité, choisissez laAWS KMS clé pour protéger les informations de la chaîne. La clé par défaut est fournie par Amazon Lex V2.

   c. Pour la configuration de l'intégration, attribuez un nom au canal et une description facultative. Choisissez l'alias qui pointe vers la version du bot à utiliser et choisissez la langue prise en charge par la chaîne.

> **Note**
>
> Si votre bot est disponible en plusieurs langues, vous devez créer une chaîne et une application différentes pour chaque langue.

d. Pour Configuration supplémentaire, entrez ce qui suit :

- ID client : saisissez l'identifiant client de Slack.
- Secret client : saisissez le secret client de Slack.
- Jeton de vérification : saisissez le jeton de vérification fourni par Slack.
- URL de la page de réussite : URL de la page que Slack doit ouvrir lorsque l'utilisateur est authentifié. En règle générale, vous laissez ce champ vide.

5. Choisissez Créer pour créer la chaîne.
6. Amazon Lex V2 affiche la liste des chaînes de votre bot. Dans la liste, choisissez la chaîne que vous venez de créer.
7. À partir de l'URL de rappel, enregistrez le point de terminaison et le point de terminaison OAuth.

Étape suivante

Étape 4 : Intégration complète à Slack

Étape 4 : Intégration complète à Slack

Dans cette section, utilisez la console d'API Slack pour terminer l'intégration à l'application Slack.

1. Connectez-vous à la console de l'API Slack à l'adresse https://api.slack.com. Choisissez l'application que vous avez créée dans Étape 2 : créer une application Slack.
2. Mettez à jour la fonction OAuth & Permissions comme suit :

   a. Dans le menu de gauche, choisissez OAuth & Permissions.

   b. Dans la section URL de redirection, ajoutez le point de terminaison OAuth fourni par Amazon Lex à l'étape précédente. Choisissez Ajouter, puis Save URLs.

   c. Dans la section Bot Token Scopes, ajoutez deux autorisations à l'aide du bouton Ajouter une étendue OAuth. Filtrez la liste avec le texte suivant :

   - **`chat:write`**

- **`team:read`**

3. Mettez à jour la fonctionnalité Interactivité et raccourcis en mettant à jour la valeur de l'URL de demande vers le point de terminaison indiqué par Amazon Lex à l'étape précédente. Entrez le point de terminaison que vous avez enregistré à l'étape 3, puis choisissez Enregistrer les modifications.
4. Abonnez-vous à la fonction Abonnements aux événements comme suit :
   - Activez les événements en choisissant l'option Activé.
   - Définissez la valeur de l'URL de demande sur le point de terminaison fourni par Amazon Lex à l'étape précédente.
   - Dans la section S'abonner aux événements de robots, sélectionnez Ajouter un événement utilisateur robot et ajoutez l'événement de**`message.im`** robot pour permettre la messagerie directe entre l'utilisateur final et le bot Slack.
   - Enregistrez les modifications.
5. Activez l'envoi de messages depuis l'onglet Messages comme suit :
   - Dans le menu de gauche, choisissez App Home.
   - Dans la section Afficher les onglets, choisissez Autoriser les utilisateurs à envoyer des commandes Slash et des messages depuis l'onglet Messages.
6. Choisissez Manage Distribution sous Paramètres. Choisissez Add to Slack pour installer l'application. Si vous êtes authentifié dans plusieurs espaces de travail, choisissez d'abord l'espace de travail approprié dans le coin supérieur droit de la liste déroulante. Sélectionnez ensuite Autoriser pour autoriser le bot à répondre aux messages.

> **Note**
>
> Si vous modifiez ultérieurement les paramètres de votre application Slack, vous devez recommencer cette sous-étape.

Étape suivante

Étape 5 : Test de l'intégration

Étape 5 : Test de l'intégration

Utilisez maintenant une fenêtre de navigateur pour tester l'intégration de Slack à votre robot Amazon Lex V2.

Pour tester votre application Slack

1. Lancez Slack. Dans le menu de gauche, dans la section Messages directs, choisissez votre robot. Si vous ne le voyez pas, choisissez l'icône plus (+) à côté de Direct Messages afin de le rechercher.
2. Discutez avec votre application Slack. Votre bot répond aux messages.

   Si vous avez créé le bot à l'aide deExercice 1 : créer un bot à partir d'un exemple, vous pouvez utiliser les exemples de conversations de cet exercice.

Intégration d'un bot Amazon Lex V2 à Twilio SMS

Cette rubrique fournit des instructions pour intégrer un bot Amazon Lex V2 au service de messagerie simple (SMS) Twilio. Procédez comme suit :

Rubriques

Étape 1 : Créer un compte Twilio SMS Twilio SMS

Créez un compte Twilio et prenez note des informations de compte suivantes :

- ACCOUNT SID
- AUTH TOKEN

Pour obtenir des instructions d'inscription, consultez https://www.twilio.com/console.

Étape suivante

Étape 2 : intégrer le point de terminaison du service de messagerie Twilio au bot Amazon Lex V2

Étape 2 : intégrer le point de terminaison du service de messagerie Twilio au bot Amazon Lex V2

1. Connectez-vous à l'AWS Management Consoleconsole Amazon Lex Amazon Lex Lex https://console.aws.amazon.com/lex/.
2. Dans la liste des robots, choisissez le robot Amazon Lex V2 que vous avez créé.
3. Dans le menu de gauche, choisissez Intégrations de chaînes, puis choisissez Ajouter une chaîne.
4. Dans Créer une chaîne, procédez comme suit :
   a. Pour Platform, choisissez Twilio.
   b. Dans Politiques d'identité, choisissez laAWS KMS clé pour protéger les informations de la chaîne. La clé par défaut Amazon Lex V2.
   c. Pour la configuration de l'intégration, attribuez un nom au canal et une description facultative. Choisissez l'alias qui pointe vers la version du bot à utiliser, puis choisissez la langue prise en charge par la chaîne.
   d. Pour une configuration supplémentaire, entrez le SID du compte et le jeton d'authentification depuis le tableau de bord Twilio.
5. Sélectionnez Create (Créer).
6. Dans la liste des chaînes, choisissez la chaîne que vous venez de créer.
7. Copiez l'URL de rappel.

Étape suivante

Étape 3 : Intégration complète de Twilio

Étape 3 : Intégration complète de Twilio

Utilisez la console Twilio pour terminer l'intégration de votre robot Amazon Lex V2 à Twilio SMS.

1. Ouvrez la console Twilio à l'adresse https://www.twilio.com/console.
2. Dans le menu de gauche, choisissez Tous les produits et services, puis choisissez Numéro de téléphone.
3. Si vous avez un numéro de téléphone, choisissez-le. Si vous n'avez pas de numéro de téléphone, choisissez Acheter un numéro pour en obtenir un.

4. Dans la section Messagerie, dans UN MESSAGE ARRIVE, entrez l'URL de rappel depuis la console Amazon Lex V2.
5. Choisissez Save (Enregistrer).

Étape suivante

Étape 4 : Tester l'intégration

Étape 4 : Tester l'intégration

Utilisez votre téléphone portable pour tester l'intégration entre SMS Twilio et votre bot. A l'aide de votre téléphone portable, envoyez des messages au numéro Twilio.

Si vous avez créé le bot à l'aide deExercice 1 : créer un bot à partir d'un exemple, vous pouvez utiliser les exemples de conversations de cet exercice.

Intégration d'un bot Amazon Lex V2 à un centre de contact

Vous pouvez intégrer les robots Amazon Lex V2 à vos centres de contact pour permettre des cas d'utilisation en libre-service à l'aide de l'API de streaming Amazon Lex V2. Utilisez ces robots comme agents de réponse vocale interactive (IVR) sur le téléphone ou comme chatbot textuel intégré à votre centre d'appels. Pour plus d'informations sur les API de streaming, consultezStreaming vers un bot Amazon Lex V2.

Les API de streaming vous permettent d'activer les fonctionnalités suivantes :

- Interruptions (« barge-in ») — Les appelants peuvent interrompre le bot et répondre à une question avant que le message ne soit terminé. Pour de plus amples informations, veuillez consulter Permettre à votre bot d'être interrompu par votre utilisateur.
- Attendre et continuer : les appelants peuvent demander au bot d'attendre s'ils ont besoin de temps pour récupérer des informations supplémentaires pendant un appel, telles qu'un numéro de carte de crédit ou un numéro de réservation. Pour de plus amples informations, veuillez consulter Permettre au bot d'attendre que l'utilisateur fournisse plus d'informations.
- Support DTMF — Les appelants peuvent fournir des informations par voie vocale ou DTMF de manière interchangeable.
- Support SSML — Vous pouvez configurer les invites du bot Amazon Lex V2 à l'aide de balises SSML pour mieux contrôler la génération de voix à partir de texte. Pour plus d'informations,

consultez la section Génération de discours à partir de documents SSML dans le guide du développeur Amazon Polly.

- Délais configurables : vous pouvez configurer le temps d'attente avant que les clients aient fini de parler avant qu'Amazon Lex V2 collecte leur saisie vocale, par exemple en répondant à une question par oui ou par non, ou en fournissant une date ou un numéro de carte de crédit. Pour de plus amples informations, veuillez consulter Configuration des délais pour la capture des données saisies par l'utilisateur.
- Mises à jour sur l'état d'avancement de l'expédition : vous pouvez configurer le bot pour qu'il réponde à plusieurs messages en fonction de l'état d'exécution de la logique métier pour la réalisation des intentions. Vous pouvez configurer le bot pour qu'il réponde par des messages au début et à la fin de l'exécution, et qu'il fournisse des mises à jour périodiques pour les fonctions Lambda de longue durée. Pour de plus amples informations, veuillez consulter Configuration des mises à jour de progression du traitement.

Kit SDK Amazon Chime

Utilisez le SDK Amazon Chime pour ajouter des fonctionnalités audio, vidéo, de partage d'écran et de messagerie en temps réel à vos applications Web ou mobiles. Le SDK Amazon Chime fournit un service audio sur le réseau téléphonique public commuté (PSTN) qui vous permet de créer des applications téléphoniques personnalisées dotées d'uneAWS Lambda fonction.

Le système audio PSTN Amazon Chime est intégré à Amazon Lex V2. Vous pouvez utiliser cette intégration pour accéder aux robots Amazon Lex V2 en tant que systèmes de réponse vocale interactive (IVR) dans les centres de contact pour les interactions audio. Utilisez-le pour intégrer Amazon Lex V2 à l'aide de services audio PSTN dans les scénarios suivants.

Intégrations aux centres d'appels : vous pouvez utiliser le connecteur vocal Amazon Chime et le service audio PSTN Amazon Chime SDK pour accéder aux robots Amazon Lex V2. Utilisez-les dans n'importe quelle application de centre de contact utilisant le protocole d'initiation de session (SIP) pour les communications vocales. Cette intégration ajoute des expériences de conversation vocale en langage naturel à votre centre de contact existant sur site ou basé sur le cloud avec support SIP. Pour obtenir la liste des plateformes de centre d'appels prises en charge, consultez les ressources relatives aux connecteurs vocaux Amazon Chime.

Le schéma suivant montre l'intégration entre un centre de contact utilisant SIP et Amazon Lex V2.

Assistance téléphonique directe : vous pouvez créer des solutions IVR personnalisées pour accéder directement aux robots Amazon Lex V2 à l'aide d'un numéro de téléphone fourni dans le SDK Amazon Chime.

Pour de plus amples informations, veuillez consulter les rubriques suivantes dans le Amazon Chime.

- Intégration SIP à l'aide d'un connecteur vocal Amazon Chime
- Utilisation du service audio PSTN Amazon Chime SDK
- Intégration du son PSTN Amazon Chime à Amazon Lex V2

Lorsque le SDK Amazon Chime envoie une demande à Amazon Lex V2, il inclut des informations spécifiques à la plateforme dans votre fonction Lambda et vos journaux de conversation. Utilisez ces informations pour déterminer l'application du centre de contact qui envoie du trafic à votre bot.

Attribut de demande de demande	Valeur
x-amz-lex:channels:plateforme	`Amazon Chime SDK PSTN Audio`

Amazon Connect

Amazon Connect est un centre de contact omnicanal dans le cloud. Vous pouvez configurer un centre de contact en quelques étapes, ajouter des agents situés n'importe où et commencer à interagir avec vos clients. Pour plus d'informations, consultez Commencer à utiliser Amazon Connect dans le guide de l'administrateur Amazon Connect.

Vous pouvez créer des expériences personnalisées pour vos clients à l'aide de communications omnicanales. Par exemple, vous pouvez proposer le chat et le contact vocal en fonction des préférences du client et des temps d'attente estimés. Pendant ce temps, les agents peuvent gérer tous les clients à partir d'une seule interface. Par exemple, ils peuvent échanger par chat avec les clients et créer des tâches ou y répondre au fur et à mesure qu'elles leur sont acheminées.

Vous pouvez utiliser Amazon Connect pour les interactions audio avec vos clients, ou Amazon Connect Chat pour les interactions sous forme de texte uniquement.

Pour plus d'informations, consultez les rubriques suivantes dans le guide de l'administrateur Amazon Connect.

- Qu'est-ce qu'Amazon Connect
- Ajouter un bot Amazon Lex V2
- Amazon Connect : obtenir le bloc de contacts saisi par le client

Lorsqu'un centre de contact envoie une demande à Amazon Lex V2, il inclut des informations spécifiques à la plate-forme sous forme d'attribut de demande pour votre fonction Lambda et vos journaux de conversation. Utilisez ces informations pour déterminer quelle application de centre de contact envoie du trafic à votre bot.

Attribut de demande commun

Attribut	Valeur
x-amz-lex:canaux:plateforme	L'une des valeurs suivantes : • `Connect` • `Connect Chat`

Cloud Genesys

Genesys Cloud est une suite de services cloud pour la communication d'entreprise, la collaboration et la gestion des centres de contact. Genesys Cloud repose sur AWS et utilise un environnement cloud distribué qui fournit un accès sécurisé aux organisations dans le monde du travail.

Pour plus d'informations, consultez les pages suivantes sur le site Web de Genesys Cloud.

- À propos du centre de contact Genesys Cloud
- À propos de l'intégration avec Amazon Lex V2

Lorsqu'un centre de contact envoie une demande à Amazon Lex V2, il inclut des informations spécifiques à la plate-forme sous forme d'attribut de demande pour votre fonction Lambda et vos journaux de conversation. Utilisez ces informations pour déterminer quelle application de centre de contact envoie du trafic à votre bot.

Attribut de demande commun

Attribut	Valeur
x-amz-lex:canaux:plateforme	• `Genesys Cloud`

En savoir plus

- Dynamisez votre centre d'appels avec Amazon Lex et Genesys Cloud

Gestion des conversations

Après avoir créé un bot, vous intégrez votre application cliente aux opérations d'exécution d'Amazon Lex V2 pour mener des conversations avec votre bot.

Lorsqu'un utilisateur entame une conversation avec votre bot, Amazon Lex V2 crée une session. Une session encapsule les informations échangées entre votre application et le bot. Pour plus d'informations, veuillez consulter Gestion des sessions avec l'API Amazon Lex V2.

Une conversation classique implique un va-et-vient entre l'utilisateur et un bot. Par exemple :

```
User : I'd like to make an appointment
Bot : What type of appointment would you like to schedule?
User : dental
Bot : When should I schedule your dental appointment?
User : Tomorrow
Bot : At what time do you want to schedule the dental appointment on 2021-01-01?
User : 9 am
Bot : 09:00 is available, should I go ahead and book your appointment?
User : Yes
Bot : Thank you. Your appointment has been set successfully.
```

Lorsque vous utilisez l'RecognizeUtteranceopération RecognizeTextor, vous devez gérer la conversation dans votre application cliente. Lorsque vous utilisez cette StartConversationopération, Amazon Lex V2 gère la conversation à votre place.

Pour gérer la conversation, vous devez envoyer les déclarations de l'utilisateur au bot jusqu'à ce que la conversation atteigne une fin logique. La conversation en cours est capturée dans l'état de session. L'état de la session est mis à jour après chaque déclaration de l'utilisateur. L'état de la session contient l'état actuel de la conversation et est renvoyé par le bot dans une réponse à chaque déclaration de l'utilisateur.

Une conversation peut prendre l'un des états suivants :

- ElicitIntent— Indique que le bot n'a pas encore déterminé l'intention de l'utilisateur.
- ElicitSlot— Indique que le bot a détecté l'intention de l'utilisateur et recueille les informations requises pour y parvenir.
- ConfirmIntent— Indique que le bot attend que l'utilisateur confirme que les informations collectées sont correctes.

- Fermé — Indique que l'intention de l'utilisateur est complète et que la conversation avec le bot a atteint une fin logique.

Un utilisateur peut spécifier une nouvelle intention une fois la première intention terminée. Pour plus d'informations, veuillez consulter Gestion du contexte de conversation.

Une intention peut avoir l'un des états suivants :

- InProgress— Indique que le bot collecte les informations nécessaires pour atteindre son objectif. Ceci est associé à l'état de la `ElicitSlot` conversation.
- En attente — Indique que l'utilisateur a demandé au robot d'attendre lorsque celui-ci a demandé des informations pour un créneau spécifique.
- Exécuté : indique que la logique métier d'une fonction Lambda associée à l'intention s'est correctement exécutée.
- ReadyForFulfillment— Indique que le bot a collecté toutes les informations requises pour atteindre l'objectif et que l'application cliente peut exécuter la logique métier de traitement des commandes.
- Échec : indique qu'une intention a échoué.

Consultez les rubriques suivantes pour savoir comment utiliser les API Amazon Lex V2 pour gérer le contexte de conversation et les sessions entre votre bot et les utilisateurs.

Rubriques

- Gestion du contexte de conversation
- Gestion des sessions avec l'API Amazon Lex V2

Gestion du contexte de conversation

Le contexte de conversation est une information que l'utilisateur, votre application cliente ou une fonction Lambda fournit à un bot Amazon Lex pour répondre à une intention. Le contexte de conversation inclut les données d'emplacement fournies par l'utilisateur, les attributs de demande définis par l'application cliente et les attributs de session créés par l'application cliente et les fonctions Lambda.

Rubriques

- Définition du contexte de l'intention

- Utilisation des valeurs d'emplacement par défaut
- Définition des attributs de session
- Définition des attributs de demande
- Régler le délai d'expiration de la session
- Partage d'informations entre les intentions
- Définition d'attributs complexes

Définition du contexte de l'intention

Amazon Lex peut déclencher des intentions en fonction du contexte. Un contexte est une variable d'état qui peut être associée à une intention lorsque vous définissez un bot. Vous configurez les contextes d'une intention lorsque vous créez l'intention à l'aide de la console ou à l'aide de l'CreateIntentopération. Vous ne pouvez utiliser le contexte que dans les paramètres régionaux anglais (États-Unis) (en-US).

Il existe deux types de relations pour les contextes : les contextes de sortie et les contextes d'entrée. Un contexte de sortie devient actif lorsqu'une intention associée est satisfaite. Un contexte de sortie est renvoyé à votre application dans la réponse de l'RecognizeUtteranceopération RecognizeTextor, et il est défini pour la session en cours. Une fois qu'un contexte est activé, il reste actif pendant le nombre de tours ou la limite de temps configurée lors de la définition du contexte.

Un contexte d'entrée spécifie les conditions dans lesquelles une intention peut être reconnue. Une intention ne peut être reconnue au cours d'une conversation que lorsque tous ses contextes de saisie sont actifs. Une intention sans contexte de saisie est toujours éligible à la reconnaissance.

Amazon Lex gère automatiquement le cycle de vie des contextes qui sont activés en concrétisant les intentions à l'aide de contextes de sortie. Vous pouvez également définir des contextes actifs lors d'un appel à l'`RecognizeUtterance`opération `RecognizeText` or.

Vous pouvez également définir le contexte d'une conversation à l'aide de la fonction Lambda pour définir l'intention. Le contexte de sortie d'Amazon Lex est envoyé à l'événement d'entrée de la fonction Lambda. La fonction Lambda peut envoyer des contextes dans sa réponse. Pour plus d'informations, veuillez consulter Activation d'une logique personnalisée avec des AWS Lambda fonctions.

Supposons, par exemple, que vous ayez l'intention de réserver une voiture de location configurée pour renvoyer un contexte de sortie appelé « book_car_filled ». Lorsque l'intention est satisfaite,

Amazon Lex définit la variable de contexte de sortie « book_car_filled ». Étant donné que « book_car_filled » est un contexte actif, une intention dont le contexte « book_car_filled » est défini comme contexte d'entrée est désormais prise en compte pour reconnaissance, à condition que l'énoncé d'un utilisateur soit reconnu comme une tentative d'obtenir cette intention. Vous pouvez l'utiliser à des fins qui n'ont de sens qu'après la réservation d'une voiture, par exemple pour envoyer un reçu par e-mail ou pour modifier une réservation.

Contexte de sortie

Amazon Lex active les contextes de sortie d'une intention lorsque celle-ci est satisfaite. Vous pouvez utiliser le contexte de sortie pour contrôler les intentions susceptibles de donner suite à l'intention actuelle.

Chaque contexte possède une liste de paramètres qui sont conservés dans la session. Les paramètres sont les valeurs de créneau correspondant à l'intention atteinte. Vous pouvez utiliser ces paramètres pour prérenseigner les valeurs des emplacements à d'autres fins. Pour plus d'informations, consultez Utilisation des valeurs d'emplacement par défaut.

Vous configurez le contexte de sortie lorsque vous créez une intention à l'aide de la console ou de l'CreateIntentopération. Vous pouvez configurer une intention avec plusieurs contextes de sortie. Lorsque l'intention est satisfaite, tous les contextes de sortie sont activés et renvoyés dans la RecognizeUtteranceréponse RecognizeTextor.

Lorsque vous définissez un contexte de sortie, vous définissez également sa durée de vie, la durée ou le nombre de tours pendant lesquels le contexte est inclus dans les réponses d'Amazon Lex. Un tour correspond à une requête envoyée par votre application à Amazon Lex. Une fois le nombre de tours ou le temps écoulé, le contexte n'est plus actif.

Votre application peut utiliser le contexte de sortie selon ses besoins. Par exemple, votre application peut utiliser le contexte de sortie pour :

- Modifiez le comportement de l'application en fonction du contexte. Par exemple, une application de voyage peut avoir une action différente pour le contexte « book_car_filled » et « rental_hotel_filled ».
- Renvoie le contexte de sortie à Amazon Lex en tant que contexte d'entrée pour l'énoncé suivant. Si Amazon Lex reconnaît que l'énoncé constitue une tentative de susciter une intention, il utilise le contexte pour limiter les intentions pouvant être renvoyées à celles correspondant au contexte spécifié.

Contexte de saisie

Vous définissez un contexte de saisie pour limiter les points de la conversation où l'intention est reconnue. Les intentions sans contexte de saisie peuvent toujours être reconnues.

Vous définissez les contextes de saisie auxquels répond une intention à l'aide de la console ou de l'`CreateIntent`opération. Une intention peut avoir plusieurs contextes d'entrée.

Pour une intention comportant plusieurs contextes d'entrée, tous les contextes doivent être actifs pour déclencher l'intention. Vous pouvez définir un contexte de saisie lorsque vous appelez l'PutSessionopération RecognizeTextRecognizeUtterance,, ou.

Vous pouvez configurer les emplacements dans le but de prendre les valeurs par défaut du contexte actif actuel. Les valeurs par défaut sont utilisées lorsqu'Amazon Lex reconnaît une nouvelle intention mais ne reçoit pas de valeur de créneau. Vous spécifiez le nom du contexte et le nom de l'emplacement dans le formulaire `#context-name.parameter-name` lorsque vous définissez l'emplacement. Pour plus d'informations, veuillez consulter Utilisation des valeurs d'emplacement par défaut.

Utilisation des valeurs d'emplacement par défaut

Lorsque vous utilisez une valeur par défaut, vous spécifiez la source d'une valeur d'emplacement à remplir pour de nouvelles intentions lorsqu'aucun emplacement n'est fourni par l'utilisateur. Cette source peut être une boîte de dialogue précédente, des attributs de demande ou de session, ou une valeur fixe que vous avez définie au moment de la création.

Vous pouvez utiliser la source suivante pour vos valeurs par défaut.

- Boîte de dialogue précédente (contextes) — #context -name.parameter-name
- Attributs de session — [attribute-name]
- Attributs de la demande — <attribute-name>
- Valeur fixe — Toute valeur qui ne correspond pas à la précédente

Lorsque vous utilisez cette CreateIntentopération pour ajouter des créneaux à une intention, vous pouvez ajouter une liste de valeurs par défaut. Les valeurs par défaut sont utilisées dans l'ordre dans lequel elles sont répertoriées. Supposons, par exemple, que vous ayez une intention avec un créneau répondant à la définition suivante :

```
"slots": [
    {
        "botId": "string",
        "defaultValueSpec": {
            "defaultValueList": [
                {
                    "defaultValue": "#book-car-fulfilled.startDate"
                },
                {
                    "defaultValue": "[reservationStartDate]"
                }
            ]
        },
        Other slot configuration settings
    }
]
```

Lorsque l'intention est reconnue, la valeur de l'emplacement nommé reservation-start-date « » est définie sur l'une des valeurs suivantes.

1. Si le contexte book-car-fulfilled « » est actif, la valeur du paramètre « StartDate » est utilisée comme valeur par défaut.
2. Si le contexte « book-car-fulfilled » n'est pas actif ou si le paramètre « StartDate » n'est pas défini, la valeur de l'attribut de session reservationStartDate « » est utilisée comme valeur par défaut.
3. Si aucune des deux premières valeurs par défaut n'est utilisée, cela signifie que l'emplacement n'a pas de valeur par défaut et Amazon Lex va obtenir une valeur comme d'habitude.

Si une valeur par défaut est utilisée pour le slot, celui-ci n'est pas activé même s'il est requis.

Définition des attributs de session

Les attributs de session contiennent des informations spécifiques à l'application qui sont transmises entre un bot et une application cliente au cours d'une session. Amazon Lex transmet les attributs de session à toutes les fonctions Lambda configurées pour un bot. Si une fonction Lambda ajoute ou met à jour des attributs de session, Amazon Lex transmet les nouvelles informations à l'application cliente.

Utilisez les attributs de session dans vos fonctions Lambda pour initialiser un bot et personnaliser les invites et les cartes-réponses. Par exemple :

- Initialisation : dans un robot de commande de pizzas, l'application cliente transmet la position de l'utilisateur en tant qu'attribut de session lors du premier appel à l'RecognizeUtteranceopération RecognizeTextou. Par exemple, `"Location": "111 Maple Street"`. La fonction Lambda utilise ces informations pour trouver la pizzeria la plus proche pour passer la commande.
- Personnalisez les invites : configurez les invites et les cartes de réponse pour faire référence aux attributs de session. Par exemple, « Hey [FirstName], quelles garnitures aimerais-tu ? » Si vous transmettez le prénom de l'utilisateur en tant qu'attribut de session (`{"FirstName": "Vivian"}`), Amazon Lex remplace l'espace réservé par le nom. Il envoie ensuite un message personnalisé à l'utilisateur : « Hey Vivian, quelles garnitures aimerais-tu ? »

Les attributs de session sont conservés pendant toute la durée de la session. Amazon Lex les stocke dans un magasin de données chiffré jusqu'à la fin de la session. Le client peut créer des attributs de session dans une demande en appelant l'RecognizeUtteranceopération RecognizeTextou avec le `sessionAttributes` champ défini sur une valeur. Une fonction Lambda peut créer un attribut de session dans une réponse. Une fois que le client ou une fonction Lambda a créé un attribut de session, la valeur de l'attribut stocké est utilisée chaque fois que l'application cliente n'inclut pas de `sessionAttribute` champ dans une demande adressée à Amazon Lex.

Par exemple, supposons que vous ayez deux attributs de session, `{"x": "1", "y": "2"}`. Si le client appelle l'`RecognizeUtterance`opération `RecognizeText` or sans spécifier le `sessionAttributes` champ, Amazon Lex appelle la fonction Lambda avec les attributs de session stockés (`{"x": 1, "y": 2}`. Si la fonction Lambda ne renvoie pas d'attributs de session, Amazon Lex renvoie les attributs de session stockés à l'application cliente.

Si l'application cliente ou une fonction Lambda transmet des attributs de session, Amazon Lex met à jour les attributs de session stockés. La transmission d'une valeur existante comme `{"x": 2}` met à jour la valeur stockée. Si vous transmettez un nouvel ensemble d'attributs de session, par exemple `{"z": 3}`, les valeurs existantes sont supprimées et seule la nouvelle valeur est conservée. Lorsqu'une carte vide, `{}`, est transmise, les valeurs stockées sont effacées.

Pour envoyer des attributs de session à Amazon Lex, vous créez une string-to-string carte des attributs. L'exemple suivant montre comment mapper des attributs de session :

```
{
   "attributeName": "attributeValue",
   "attributeName": "attributeValue"
}
```

Pour l'`RecognizeText`opération, vous insérez la carte dans le corps de la requête à l'aide du `sessionAttributes` champ de la `sessionState` structure, comme suit :

```
"sessionState": {
    "sessionAttributes": {
        "attributeName": "attributeValue",
        "attributeName": "attributeValue"
    }
}
```

Pour cette `RecognizeUtterance` opération, vous codez la carte en base64, puis vous l'envoyez dans le cadre de l'`x-amz-lex-session-state`en-tête.

Si vous envoyez des données structurées ou binaires dans un attribut de session, vous devez tout d'abord convertir les données en chaîne simple. Pour plus d'informations, veuillez consulter Définition d'attributs complexes.

Définition des attributs de demande

Les attributs de demandes contiennent des informations spécifiques à la demande et s'appliquent uniquement à la demande en cours. Une application cliente envoie ces informations à Amazon Lex. Utilisez les attributs de demande pour transmettre des informations qui n'ont pas besoin de persister pendant la totalité de la session. Vous pouvez utiliser vos propres attributs de demandes ou des attributs prédéfinis. Pour envoyer des attributs de demandes, utilisez l'en-tête `x-amz-lex-request-attributes` dans RecognizeUtterance ou le champ `requestAttributes` dans une demande RecognizeText. Dans la mesure où les attributs de demandes ne sont pas conservés entre toutes les demandes (contrairement aux attributs de session), ils ne sont pas renvoyés dans les réponses `RecognizeUtterance` ou `RecognizeText`.

> **Note**
>
> Pour envoyer des informations qui persistent entre les demandes, utilisez des attributs de session.

Définition des attributs de demande définis par l'utilisateur

Un attribut de demande défini par l'utilisateur correspond à des données que vous envoyez au bot dans chaque demande. Vous envoyez les informations dans l'en-tête `amz-lex-request-`

`attributes` d'une demande `RecognizeUtterance` ou dans le champ `requestAttributes` d'une demande `RecognizeText`.

Pour envoyer des attributs de demande à Amazon Lex, vous créez une string-to-string carte des attributs. L'exemple suivant montre comment mapper des attributs de demandes :

```
{
   "attributeName": "attributeValue",
   "attributeName": "attributeValue"
}
```

Pour l'opération `PostText`, vous insérez le mappage dans le corps de la requête en utilisant le champ `requestAttributes` comme suit :

```
"requestAttributes": {
   "attributeName": "attributeValue",
   "attributeName": "attributeValue"
}
```

Pour l'opération `PostContent`, vous codez le mappage en base64, puis l'envoyez en tant qu'en-tête `x-amz-lex-request-attributes`.

Si vous envoyez des données structurées ou binaires dans un attribut de demande, vous devez tout d'abord convertir les données en chaîne simple. Pour plus d'informations, veuillez consulter Définition d'attributs complexes.

Régler le délai d'expiration de la session

Amazon Lex conserve les informations contextuelles (données des emplacements et attributs de session) jusqu'à la fin d'une session de conversation. Pour contrôler la durée d'une session pour un bot, définissez le délai d'expiration de la session. Par défaut, la durée de la session est de 5 minutes, mais vous pouvez spécifier n'importe quelle durée comprise entre 0 et 1 440 minutes (24 heures).

Supposons que vous créiez un bot `ShoeOrdering` qui prend en charge des intentions comme `OrderShoes` et `GetOrderStatus`. Lorsqu'Amazon Lex détecte que l'intention de l'utilisateur est de commander des chaussures, il lui demande des données sur les créneaux. Par exemple, il demande la pointure, la couleur, la marque, etc. Si l'utilisateur fournit certaines données relatives à l'emplacement mais ne termine pas l'achat de la chaussure, Amazon Lex mémorise toutes les données relatives à l'emplacement et les attributs de session pendant toute la session. Si l'utilisateur

revient à la session avant son expiration, il peut fournir les données d'emplacement restantes et terminer l'achat.

Dans la console Amazon Lex, vous définissez le délai d'expiration de la session lorsque vous créez un bot. À l'aide de l'interface de ligne de commande AWS (CLI AWS) ou de l'API, vous définissez le délai d'expiration lorsque vous créez un bot avec l'CreateBotopération en définissant le champ InSecondsIdleSessionTTL.

Partage d'informations entre les intentions

Amazon Lex prend en charge le partage d'informations entre les intentions. Pour partager des intentions, utilisez des contextes de sortie ou des attributs de session.

Pour utiliser des contextes de sortie, vous devez définir un contexte de sortie lorsque vous créez ou mettez à jour une intention. Lorsque l'intention est satisfaite, les réponses d'Amazon Lex V2 contiennent le contexte et les valeurs de créneau de l'intention sous forme de paramètres contextuels. Vous pouvez utiliser ces paramètres comme valeurs par défaut dans les intentions suivantes ou dans le code de votre application ou dans les fonctions Lambda.

Pour utiliser les attributs de session, vous devez les définir dans votre code Lambda ou dans votre code d'application. Supposons qu'un utilisateur du bot `ShoeOrdering` commence par commander des chaussures. Le bot engage une conversation avec l'utilisateur, en collectant des données d'option telles que la pointure, la couleur et la marque. Lorsque l'utilisateur passe une commande, la fonction Lambda qui exécute la commande définit l'attribut `orderNumber` session, qui contient le numéro de commande. Pour obtenir le statut de la commande, l'utilisateur utilise l'intention `GetOrderStatus`. Le bot peut demander à l'utilisateur des données d'option, comme le numéro et la date de commande. Lorsqu'il reçoit les informations requises, il renvoie le statut de la commande.

Si vous pensez que vos utilisateurs peuvent changer d'intention au cours de la même session, vous pouvez concevoir le bot pour qu'il renvoie le statut de la dernière commande. Au lieu de redemander à l'utilisateur des informations sur sa commande, vous utilisez l'attribut de session `orderNumber` pour partager les informations entre les intentions et traiter l'intention `GetOrderStatus`. Le bot effectue cette opération en renvoyant le statut de la dernière commande passée par l'utilisateur.

Définition d'attributs complexes

Les attributs de session et de demande sont string-to-string des cartes d'attributs et de valeurs. Dans de nombreux cas, vous pouvez utiliser le mappage de chaînes pour transférer les valeurs d'attribut entre votre application cliente et un bot. Cependant, dans certains cas, vous devez transférer des

données binaires ou une structure complexe qu'il n'est pas facile de convertir en mappage de chaînes. Par exemple, l'objet JSON suivant représente un tableau des trois villes les plus peuplées aux Etats-Unis :

```
{
   "cities": [
      {
         "city": {
            "name": "New York",
            "state": "New York",
            "pop": "8537673"
         }
      },
      {
         "city": {
            "name": "Los Angeles",
            "state": "California",
            "pop": "3976322"
         }
      },
      {
         "city": {
            "name": "Chicago",
            "state": "Illinois",
            "pop": "2704958"
         }
      }
   ]
}
```

Ce tableau de données ne se traduit pas bien en string-to-string carte. Dans ce cas, vous pouvez convertir un objet en chaîne simple pour pouvoir l'envoyer au bot avec les opérations RecognizeText et RecognizeUtterance.

Par exemple, si vous utilisezJavaScript, vous pouvez utiliser l'`JSON.stringify`opération pour convertir un objet en JSON et l'`JSON.parse`opération pour convertir le texte JSON en JavaScript objet :

```
// To convert an object to a string.
var jsonString = JSON.stringify(object, null, 2);
// To convert a string to an object.
```

```
var obj = JSON.parse(JSON string);
```

Pour envoyer des attributs avec l'`RecognizeUtterance`opération, vous devez les encoder en base64 avant de les ajouter à l'en-tête de la demande, comme indiqué dans le code suivant : JavaScript

```
var encodedAttributes = new Buffer(attributeString).toString("base64");
```

Pour envoyer des données binaires aux opérations `RecognizeText` et `RecognizeUtterance`, convertissez d'abord les données en chaîne codée en base64, puis envoyez la chaîne en tant que valeur dans les attributs de session :

```
"sessionAttributes" : {
   "binaryData": "base64 encoded data"
}
```

Gestion des sessions avec l'API Amazon Lex V2

Lorsqu'un utilisateur entame une conversation avec votre bot, Amazon Lex V2 crée une session. Les informations échangées entre votre application et Amazon Lex V2 constituent l'état de session de la conversation. Lorsque vous faites une demande, la session est identifiée par un identifiant que vous spécifiez. Pour plus d'informations sur l'identifiant de session, consultez le `sessionId` champ de l'RecognizeUtteranceopération RecognizeTextor.

Vous pouvez modifier l'état de session envoyé entre votre application et votre bot. Par exemple, vous pouvez créer et modifier des attributs de session qui contiennent des informations personnalisées sur la session, et vous pouvez modifier le flux de la conversation en définissant le contexte de dialogue pour interpréter le prochain énoncé.

Vous pouvez mettre à jour l'état d'une session de trois manières.

- Transmettez les informations de session en ligne dans le cadre d'un appel à l'`RecognizeUtterance`opération `RecognizeText` or.
- Utilisez une fonction Lambda avec l'`RecognizeUtterance`opération `RecognizeText` ou qui est appelée après chaque tour de conversation. Pour plus d'informations, veuillez consulter Activation d'une logique personnalisée avec des AWS Lambda fonctions. L'autre consiste à utiliser l'API d'exécution Amazon Lex V2 dans votre application pour modifier l'état de la session.

- Utilisez des opérations qui vous permettent de gérer les informations de session pour une conversation avec votre bot. Les opérations sont l'PutSessionopération, l'GetSessionopération et l'DeleteSessionopération. Vous utilisez ces opérations pour obtenir des informations sur l'état de session de votre utilisateur dans votre bot et avoir un contrôle précis sur l'état.

Utilisez l'opération `GetSession` lorsque vous souhaitez obtenir l'état actuel de la session. L'opération renvoie l'état actuel de la session, y compris l'état de la boîte de dialogue avec votre utilisateur, tous les attributs de session qui ont été définis, les valeurs de créneau correspondant à l'intention actuelle et toutes les autres intentions identifiées par Amazon Lex V2 comme étant des intentions possibles correspondant à l'énoncé de l'utilisateur.

L'opération `PutSession` vous permet de manipuler directement l'état de session en cours. Vous pouvez définir la session, notamment le type d'action de dialogue que le bot exécutera ensuite et les messages qu'Amazon Lex V2 envoie à l'utilisateur. Cela vous permet de contrôler le flux de la conversation avec le bot. Définissez le `type` champ d'action de la boîte de dialogue `Delegate` pour qu'Amazon Lex V2 détermine l'action suivante pour le bot.

Vous pouvez utiliser l'opération `PutSession` pour créer une nouvelle session avec un bot et définir l'intention avec laquelle le bot doit démarrer. Vous pouvez également utiliser l'opération `PutSession` pour passer d'une intention à une autre. Lorsque vous créez une session ou modifiez l'intention, vous pouvez également définir un état de session, comme des valeurs d'options et des attributs de session. Lorsque la nouvelle intention est terminée, vous avez la possibilité de redémarrer l'intention précédente.

La réponse générée depuis l'opération `PutSession` contient les mêmes informations que l'opération `RecognizeUtterance`. Vous pouvez utiliser ces informations pour demander à l'utilisateur l'élément d'information suivant, comme vous le feriez avec la réponse de l'opération `RecognizeUtterance`.

Utilisez l'opération `DeleteSession` pour supprimer une session existante et démarrer avec une nouvelle session. Par exemple, lorsque vous testez le bot, vous pouvez utiliser l'opération `DeleteSession` pour supprimer des sessions de test de votre bot.

Les opérations de session fonctionnent avec vos fonctions Lambda de traitement des commandes. Par exemple, si votre fonction Lambda renvoie `Failed` l'état d'exécution, vous pouvez utiliser l'`PutSession`opération pour définir le type d'action de la boîte de dialogue `close` et pour `fulfillmentState` `ReadyForFulfillment` réessayer l'étape de traitement.

Voici quelques actions que vous pouvez effectuer avec les opérations de session :

- Demander au bot de démarrer une conversation au lieu d'attendre l'utilisateur.
- Changer d'intention au cours d'une conversation.
- Revenir à une intention précédente.
- Démarrer ou redémarrer une conversation au milieu de l'interaction.
- Valider des valeurs d'option et demander au bot d'entrer à nouveau des valeurs en cas de valeurs non valides.

Chacune de ces actions sont décrites plus en détail ci-dessous.

Démarrage d'une nouvelle session

Si vous souhaitez que le bot démarre la conversation avec votre utilisateur, vous pouvez utiliser l'opération `PutSession`.

- Créez une intention de bienvenue sans options et un message de conclusion qui invite l'utilisateur à indiquer une intention. Par exemple, « Que souhaitez-vous commander ? Vous pouvez dire « Commander une boisson » ou « Commander une pizza ». »
- Appelez l'opération `PutSession`. Définissez nom de l'intention sur le nom de votre intention de bienvenue et définissez l'action de dialogue sur `Delegate`.
- Amazon Lex répondra en vous invitant à entamer la conversation avec votre utilisateur, comme suite à votre message de bienvenue.

Changer d'intention

Vous pouvez utiliser l'opération `PutSession` pour passer d'une intention à une autre. Vous pouvez également l'utiliser pour revenir à une intention précédente. Vous pouvez utiliser l'opération `PutSession` pour définir des attributs de session ou des valeurs d'option pour la nouvelle intention.

- Appelez l'opération `PutSession`. Définissez le nom sur le nom de la nouvelle intention et définissez l'action de dialogue sur `Delegate`. Vous pouvez également définir les valeurs d'options ou les attributs de session requis pour la nouvelle intention.
- Amazon Lex entamera une conversation avec l'utilisateur en utilisant la nouvelle intention.

Reprise d'une intention antérieure

Pour reprendre une intention antérieure, vous utilisez l'`GetSession`opération pour obtenir l'état de l'intention, effectuer l'interaction requise, puis utiliser l'`PutSession`opération pour rétablir l'état précédent de l'intention dans la boîte de dialogue.

- Appelez l'opération `GetSession`. Enregistrez l'état de l'intention.
- Effectuez une autre interaction, par exemple en répondant à une autre intention.
- À l'aide des informations enregistrées pour l'intention précédente, appelez l'`PutSession`opération. Cela renverra à l'utilisateur vers l'intention précédente au même endroit dans la conversation.

Dans certains cas, il peut être nécessaire de reprendre la conversation de votre utilisateur avec votre bot. Par exemple, imaginons que vous avez créé un bot de service client. Votre application détermine que l'utilisateur a besoin de parler à un représentant du service client. Après avoir parlé avec l'utilisateur, le représentant peut rediriger la conversation vers le bot avec les informations qu'il a collectées.

Pour reprendre une session, utilisez des étapes similaires aux étapes suivantes :

- Votre application détermine que l'utilisateur a besoin de parler à un représentant du service client.
- Utilisez l'opération `GetSession` pour obtenir l'état de dialogue actuel de l'intention.
- Le représentant service client parle à l'utilisateur et résout le problème.
- Utilisez l'opération `PutSession` pour définir l'état de dialogue de l'intention. Cela peut inclure la définition de valeurs d'option et d'attributs de session, ou la modification de l'intention.
- Le bot reprend la conversation avec l'utilisateur.

Validation des valeurs des emplacements

Vous pouvez valider les réponses adressées à votre bot à l'aide de votre application cliente. Si la réponse n'est pas valide, vous pouvez utiliser l'opération `PutSession` pour obtenir une nouvelle réponse de votre utilisateur. Par exemple, supposons que votre bot de commande de fleurs ne peut vendre que des tulipes, des roses et des lys. Si l'utilisateur commande des œillets, votre application peut effectuer les opérations suivantes :

- Examiner la valeur d'option renvoyée à partir de la réponse `PostText` ou `PostContent`.

- Si la valeur d'option n'est pas valide, appeler l'opération `PutSession`. Votre application doit effacer la valeur d'option, définir le champ `slotToElicit` et définir la valeur de `dialogAction.type` sur `elicitSlot`. Vous pouvez éventuellement définir les `messageFormat` champs `message` et si vous souhaitez modifier le message qu'Amazon Lex utilise pour obtenir la valeur de l'emplacement.

Activation d'une logique personnalisée avec des AWS Lambda fonctions

Grâce aux AWS Lambdafonctions, vous pouvez mieux contrôler le comportement de votre bot Amazon Lex V2 grâce à des fonctions personnalisées que vous définissez.

Amazon Lex V2 utilise une fonction Lambda par alias de bot par langue au lieu d'une fonction Lambda pour chaque intention.

Pour intégrer une fonction Lambda à votre bot Amazon Lex V2, effectuez les étapes suivantes :

1. Déterminez les champs de l'événement d'entrée dont vous souhaitez tirer des informations à utiliser dans votre fonction Lambda.
2. Déterminez les champs de la réponse que vous souhaitez manipuler et renvoyer par votre fonction Lambda.
3. Créez une fonction en AWS Lambda utilisant le langage de programmation de votre choix et rédigez votre script.
4. Assurez-vous que la fonction renvoie une structure correspondant au format de réponse.
5. Déployez la fonction Lambda.
6. Associez la fonction Lambda à un alias de bot Amazon Lex V2 aux opérations de console ou d'API.
7. Sélectionnez les étapes de conversation au cours desquelles vous souhaitez appeler votre fonction Lambda avec les opérations de console ou d'API.
8. Créez votre bot Amazon Lex V2 et vérifiez que la fonction Lambda fonctionne comme prévu. Déboguez votre fonction à l'aide d'Amazon CloudWatch.

Rubriques

Interprétation du format d'événement d'entrée

La première étape de l'intégration d'une fonction Lambda dans votre bot Amazon Lex V2 consiste à comprendre les champs de l'événement Amazon Lex V2 et à déterminer les informations de ces champs que vous souhaitez utiliser lors de l'écriture de votre script. L'objet JSON suivant montre le format général d'un événement Amazon Lex V2 transmis à une fonction Lambda :

> **Note**
>
> Le format d'entrée peut changer sans qu'une modification correspondante ne soit apportée au`messageVersion`. Votre code ne devrait pas générer d'erreur si de nouveaux champs sont présents.

```
{
    "messageVersion": "1.0",
    "invocationSource": "DialogCodeHook | FulfillmentCodeHook",
    "inputMode": "DTMF | Speech | Text",
    "responseContentType": "audio/mpeg | audio/ogg | audio/pcm | text/plain;
 charset=utf-8",
    "sessionId": string,
    "inputTranscript": string,
    "invocationLabel": string,
    "bot": {
        "id": string,
        "name": string,
        "localeId": string,
        "version": string,
        "aliasId": string,
        "aliasName": string
    },
    "interpretations": [
        {
            "interpretationSource": "Bedrock | Lex",
            "intent": {
                // see Intention for details about the structure
            },
            "nluConfidence": number,
            "sentimentResponse": {
                "sentiment": "MIXED | NEGATIVE | NEUTRAL | POSITIVE",
                "sentimentScore": {
```

```
                    "mixed": number,
                    "negative": number,
                    "neutral": number,
                    "positive": number
                }
            }
        },
        ...
    ],
    "proposedNextState": {
        "dialogAction": {
            "slotToElicit": string,
            "type": "Close | ConfirmIntent | Delegate | ElicitIntent | ElicitSlot"
        },
        "intent": {
            // see Intention for details about the structure
        },
        "prompt": {
            "attempt": string
        }
    },
    "requestAttributes": {
        string: string,
        ...
    },
    "sessionState": {
        // see État de la session for details about the structure
    },
    "transcriptions": [
        {
            "transcription": string,
            "transcriptionConfidence": number,
            "resolvedContext": {
                "intent": string
            },
            "resolvedSlots": {
                slot name: {
                    // see Emplacements for details about the structure
                },
                ...
            }
        },
        ...
    ]
```

```
}
```

Chaque champ de l'événement d'entrée est décrit ci-dessous :

messageVersion

Version du message qui identifie le format des données d'événement entrées dans la fonction Lambda et le format attendu de la réponse d'une fonction Lambda.

> **Note**
>
> Vous configurez cette valeur lorsque vous définissez une intention. Dans l'implémentation actuelle, Amazon Lex V2 ne prend en charge que la version 1.0 des messages. Par conséquent, la console prend la valeur par défaut 1.0 et n'affiche pas la version de message.

invocationSource

Le crochet de code qui a appelé la fonction Lambda. Les valeurs suivantes sont possibles :

`DialogCodeHook`— Amazon Lex V2 a appelé la fonction Lambda après une entrée de l'utilisateur.

`FulfillmentCodeHook`— Amazon Lex V2 a appelé la fonction Lambda une fois que tous les emplacements requis ont été remplis et que l'intention est prête à être exécutée.

Mode d'entrée

Mode d'énoncé de l'utilisateur. Les valeurs possibles sont les suivantes :

`DTMF`— L'utilisateur saisit l'énoncé à l'aide d'un clavier tactile (double tonalité multifréquence).

`Speech`— L'utilisateur a prononcé l'énoncé.

`Text`— L'utilisateur a saisi l'énoncé.

responseContentType

Mode de réponse du bot à l'utilisateur. `text/plain; charset=utf-8`indique que le dernier énoncé a été écrit, tandis qu'une valeur commençant par `audio` indique que le dernier énoncé a été prononcé.

sessionId

Identifiant de session alphanumérique utilisé pour la conversation.

inputTranscript

Une transcription de l'entrée de l'utilisateur.

- Pour la saisie de texte, il s'agit du texte saisi par l'utilisateur. Pour l'entrée DTMF, il s'agit de la clé saisie par l'utilisateur.
- Pour la saisie vocale, il s'agit du texte en lequel Amazon Lex V2 convertit l'énoncé de l'utilisateur afin d'invoquer une intention ou de remplir un espace.

Étiquette d'invocation

Une valeur qui indique la réponse qui a appelé la fonction Lambda. Vous pouvez définir des étiquettes d'appel pour la réponse initiale, les créneaux et la réponse de confirmation.

bot

Informations sur le bot qui a traité la demande, comprenant les champs suivants :

- id — L'identifiant attribué au bot lorsque vous l'avez créé. Vous pouvez voir l'ID du bot dans la console Amazon Lex V2 sur la page des paramètres du bot.
- name — Le nom que vous avez donné au bot lorsque vous l'avez créé.
- LocaleID — L'identifiant de la localisation que vous avez utilisée pour votre bot. Pour obtenir la liste des paramètres régionaux, voirLangues et paramètres régionaux pris en charge par Amazon Lex V2.
- version — Version du bot qui a traité la demande.
- AliasID — Identifiant attribué à l'alias du bot lorsque vous l'avez créé. Vous pouvez voir l'ID d'alias du bot dans la console Amazon Lex V2 sur la page Alias. Si vous ne voyez pas l'identifiant d'alias dans la liste, cliquez sur l'icône en forme de roue dentée en haut à droite et activez l'identifiant d'alias.
- aliasName — Le nom que vous avez donné à l'alias du bot.

interprétations

Une liste d'informations sur les intentions qu'Amazon Lex V2 considère comme possibles correspond à l'énoncé de l'utilisateur. Chaque élément est une structure qui fournit des informations sur la correspondance de l'énoncé avec une intention, au format suivant :

```
{
    "intent": {
        // see Intention for details about the structure
    },
    "interpretationSource": "Bedrock | Lex",
    "nluConfidence": number,
    "sentimentResponse": {
        "sentiment": "MIXED | NEGATIVE | NEUTRAL | POSITIVE",
        "sentimentScore": {
            "mixed": number,
            "negative": number,
            "neutral": number,
            "positive": number
        }
    }
}
```

Les champs de la structure sont les suivants :

- intention — Structure contenant des informations sur l'intention. Voir Intention pour plus de détails sur la structure.
- NLUConfidence : score qui indique dans quelle mesure Amazon Lex V2 est certain que l'intention correspond à celle de l'utilisateur.
- SentimentResponse — Analyse du sentiment de la réponse, contenant les champs suivants :
  - sentiment — Indique si le sentiment de l'énoncé est`POSITIVE`, `NEGATIVE`NEUTRAL, ou. `MIXED`
  - SentimentScore : structure associant chaque sentiment à un chiffre indiquant dans quelle mesure Amazon Lex V2 est sûr que l'énoncé exprime ce sentiment.
- InterpretationSource — Indique si un emplacement est résolu par Amazon Lex ou Amazon Bedrock.

proposedNextState

Si la fonction Lambda définit le `dialogAction` `sessionState` à`Delegate`, ce champ apparaît et indique la proposition d'Amazon Lex V2 pour l'étape suivante de la conversation. Dans le cas contraire, l'état suivant dépend des paramètres que vous renvoyez dans la réponse de votre fonction Lambda. Cette structure n'est présente que si les deux affirmations suivantes sont vraies :

1. La `invocationSource` valeur est `DialogCodeHook`
2. La prédiction `type` de `dialogAction` est`ElicitSlot`.

Vous pouvez utiliser ces informations pour les ajouter `runtimeHints` au bon moment de la conversation. Voir Amélioration de la reconnaissance des valeurs des créneaux grâce à des indices d'exécution pour plus d'informations. `proposedNextState`est une structure contenant les champs suivants :

La structure de `proposedNextState` est la suivante :

```
"proposedNextState": {
    "dialogAction": {
        "slotToElicit": string,
        "type": "Close | ConfirmIntent | Delegate | ElicitIntent | ElicitSlot"
    },
    "intent": {
        // see Intention for details about the structure
    },
    "prompt": {
        "attempt": string
    }
}
```

- DialogAction : contient des informations sur la prochaine étape proposée par Amazon Lex V2. Les champs de la structure sont les suivants :
  - slotToElicit— L'emplacement à sélectionner ensuite, tel que proposé par Amazon Lex V2. Ce champ n'apparaît que si `type` c'est le cas`ElicitSlot`.
  - type — L'étape suivante de la conversation telle que proposée par Amazon Lex V2. Les valeurs suivantes sont possibles :

    `Delegate`— Amazon Lex V2 détermine l'action suivante.

`ElicitIntent`— L'action suivante consiste à obtenir une intention de la part de l'utilisateur.

`ElicitSlot`— L'action suivante consiste à obtenir une valeur d'emplacement auprès de l'utilisateur.

`Close`— Met fin au processus de réalisation des intentions et indique qu'il n'y aura pas de réponse de la part de l'utilisateur.

`ConfirmIntent`— L'action suivante consiste à demander à l'utilisateur si les créneaux sont corrects et si l'intention est prête à être remplie.

- intention — L'intention que le bot a déterminée et que l'utilisateur essaie de réaliser. Voir Intention pour plus de détails sur la structure.
- prompt — Structure contenant le champ`attempt`, qui correspond à une valeur indiquant le nombre de fois où Amazon Lex V2 a demandé à l'utilisateur de choisir l'emplacement suivant. Les valeurs possibles concernent `Initial` la première tentative`Retry1`,,`Retry2`, `Retry3``Retry4`, et `Retry5` les tentatives suivantes.

requestAttributes

Structure contenant les attributs spécifiques à la demande que le client envoie dans la demande. Utilisez les attributs de demande pour transmettre des informations qui n'ont pas besoin de persister pendant la totalité de la session. S'il n'y a pas d'attributs de demandes, cette valeur est null. Pour en savoir plus, consultez Définition des attributs de demande.

État de la session

État actuel de la conversation entre l'utilisateur et votre bot Amazon Lex V2. Voir État de la session pour plus de détails sur la structure.

transcriptions

Une liste de transcriptions considérées comme possibles par Amazon Lex V2 correspond à l'énoncé de l'utilisateur. Pour en savoir plus, consultez Utilisation des scores de fiabilité de la transcription vocale. Chaque élément est un objet au format suivant, contenant des informations sur une transcription possible :

```
{
    "transcription": string,
```

```
    "transcriptionConfidence": number,
    "resolvedContext": {
        "intent": string
    },
    "resolvedSlots": {
        slot name: {
            // see Emplacements for details about the structure
        },
        ...
    }
}
```

Les champs sont décrits ci-dessous :

- transcription : transcription considérée par Amazon Lex V2 comme pouvant correspondre à l'énoncé audio de l'utilisateur.
- TranscriptionConfidence : score qui indique dans quelle mesure Amazon Lex V2 est certain que l'intention correspond à celle de l'utilisateur.
- ResolvedContext — Structure contenant le champ`intent`, qui correspond à l'intention à laquelle se rapporte l'énoncé.
- ResolvedSlots — Structure dont les clés sont les noms de chaque emplacement résolu par l'énoncé. Le nom de chaque emplacement correspond à une structure contenant des informations sur cet emplacement. Voir Emplacements pour plus de détails sur la structure.

Préparation du format de réponse

La deuxième étape de l'intégration d'une fonction Lambda dans votre bot Amazon Lex V2 consiste à comprendre les champs de la réponse de la fonction Lambda et à déterminer les paramètres que vous souhaitez manipuler. L'objet JSON suivant montre le format général d'une réponse Lambda renvoyée à Amazon Lex V2 :

```
{
    "sessionState": {
        // see État de la session for details about the structure
    },
    "messages": [
        {
            "contentType": "CustomPayload | ImageResponseCard | PlainText | SSML",
            "content": string,
```

```
            "imageResponseCard": {
                "title": string,
                "subtitle": string,
                "imageUrl": string,
                "buttons": [
                    {
                        "text": string,
                        "value": string
                    },
                    ...
                ]
            }
        },
        ...
    ],
    "requestAttributes": {
        string: string,
        ...
    }
}
```

Chaque champ de la réponse est décrit ci-dessous :

État de la session

État de la conversation entre l'utilisateur et votre bot Amazon Lex V2 que vous souhaitez renvoyer. Voir État de la session pour plus de détails sur la structure. Ce champ est toujours obligatoire.

messages

Liste des messages qu'Amazon Lex V2 renvoie au client pour la prochaine étape de la conversation. Si le message `contentType` que vous fournissez est `PlainTextCustomPayload`, ou`SSML`, écrivez le message que vous souhaitez renvoyer au client dans le `content` champ. Si c'est le `contentType` cas`ImageResponseCard`, donnez les détails de la carte dans le `imageResponseCard` champ. Si vous ne fournissez aucun message, Amazon Lex V2 utilise le message approprié défini lors de la création du bot.

Le `messages` champ est obligatoire s'il s'`dialogAction.type`agit de `ElicitIntent` ou`ConfirmIntent`.

Chaque élément de la liste est une structure au format suivant, contenant des informations sur un message à renvoyer à l'utilisateur. Voici un exemple :

```
{
    "contentType": "CustomPayload | ImageResponseCard | PlainText | SSML",
    "content": string,
    "imageResponseCard": {
        "title": string,
        "subtitle": string,
        "imageUrl": string,
        "buttons": [
            {
                "text": string,
                "value": string
            },
            ...
        ]
    }
}
```

Une description de chaque champ est fournie ci-dessous :

- ContentType — Type de message à utiliser.

  `CustomPayload`— Chaîne de réponse que vous pouvez personnaliser pour inclure des données ou des métadonnées pour votre application.

  `ImageResponseCard`— Une image avec des boutons que le client peut sélectionner. Voir ImageResponseCardpour plus d'informations.

  `PlainText`— Chaîne de texte brut.

  `SSML`— Chaîne qui inclut le langage de balisage de synthèse vocale pour personnaliser la réponse audio.
- content — Le message à envoyer à l'utilisateur. Utilisez ce champ si le type de message est `PlainTextCustomPayload`, ou`SSML`.
- imageResponseCard— Contient la définition de la carte-réponse à montrer à l'utilisateur. Utilisez ce champ si le type de message est`ImageResponseCard`. Correspond à une structure contenant les champs suivants :
  - title — Le titre de la carte-réponse.
  - sous-titre : invite l'utilisateur à choisir un bouton.
  - imageURL — Lien vers l'image de la carte.

- boutons : liste de structures contenant des informations sur un bouton. Chaque structure contient un `text` champ contenant le texte à afficher et un `value` champ contenant la valeur à envoyer à Amazon Lex V2 si le client sélectionne ce bouton. Vous pouvez inclure jusqu'à trois boutons.

requestAttributes

Structure contenant des attributs spécifiques à la demande pour la réponse au client. Pour plus d'informatons, consultez Définition des attributs de demande. Ce champ est facultatif.

Champs obligatoires dans la réponse

Au minimum, la réponse Lambda doit inclure `sessionState` un objet. Dans ce cadre, fournissez un `dialogAction` objet et spécifiez le `type` champ. En fonction `type` de `dialogAction` ce que vous fournissez, il se peut que d'autres champs soient obligatoires pour la réponse Lambda. Ces exigences sont décrites comme suit, accompagnées d'exemples pratiques minimaux :

Délégué

Delegate permet à Amazon Lex V2 de déterminer l'étape suivante. Aucun autre champ n'est obligatoire.

```
{
    "sessionState": {
        "dialogAction": {
            "type": "Delegate"
    }
}
```

ElicitIntent

ElicitIntentinvite le client à exprimer son intention. Vous devez inclure au moins un message dans le `messages` champ pour obtenir une indication d'intention.

```
{
    "sessionState": {
        "dialogAction": {
            "type": "ElicitIntent"
    },
    "messages": [
        {
            "contentType": PlainText,
```

```
            "content": "How can I help you?"
        }
    ]
}
```

ElicitSlot

ElicitSlotinvite le client à fournir une valeur de créneau. Vous devez inclure le nom de l'emplacement dans le `slotToElicit` champ de l'`dialogAction`objet. Vous devez également inclure le `name` de `intent` dans l'`sessionState`objet.

```
{`
    "sessionState": {
        "dialogAction": {
            "slotToElicit": "OriginCity",
            "type": "ElicitSlot"
        },
        "intent": {
            "name": "BookFlight"
        }
    }
}
```

ConfirmIntent

ConfirmIntentconfirme les valeurs des créneaux du client et confirme si l'intention est prête à être réalisée. Vous devez inclure le `name` du `intent` dans l'`sessionState`objet et le `slots` à confirmer. Vous devez également inclure au moins un message dans le `messages` champ pour demander à l'utilisateur de confirmer les valeurs des créneaux. Votre message doit recevoir une réponse « oui » ou « non ». Si l'utilisateur répond « oui », Amazon Lex V2 définit `confirmationState` l'intention sur`Confirmed`. Si l'utilisateur répond « non », Amazon Lex V2 définit `confirmationState` l'intention sur`Denied`.

```
{
    "sessionState": {
        "dialogAction": {
            "type": "ConfirmIntent"
        },
        "intent": {
            "name": "BookFlight",
            "slots": {
                "DepartureDate": {
```

```
                    "value": {
                        "originalValue": "tomorrow",
                        "interpretedValue": "2023-05-09",
                        "resolvedValues": [
                            "2023-05-09"
                        ]
                     }
                    },
                    "DestinationCity": {
                        "value": {
                            "originalValue": "sf",
                            "interpretedValue": "sf",
                            "resolvedValues": [
                                "sf"
                            ]
                        }
                    },
                    "OriginCity": {
                        "value": {
                            "originalValue": "nyc",
                            "interpretedValue": "nyc",
                            "resolvedValues": [
                                "nyc"
                            ]
                        }
                    }
                }
            }
        },
        "messages": [
            {
                "contentType": PlainText,
                "content": "Okay, you want to fly from {OriginCity} to \
                {DestinationCity} on {DepartureDate}. Is that correct?"
            }
        ]
    }
```

Fermer

La fermeture met fin au processus de réalisation de l'intention et indique qu'aucune autre réponse n'est attendue de la part de l'utilisateur. Vous devez inclure le `name` et `state` de `intent` dans l'`sessionState`objet. Les états d'intention compatibles sont `FailedFulfilled`, et`InProgress`.

```
"sessionState": {
    "dialogAction": {
        "type": "Close"
    },
    "intent": {
        "name": "BookFlight",
        "state": "Failed | Fulfilled | InProgress"
    }
}
```

Structures communes de l'événement et de la réponse Lambda

Dans la réponse Lambda, un certain nombre de structures se reproduisent. Des détails sur ces structures communes sont fournis dans cette section.

Intention

```
"intent": {
    "confirmationState": "Confirmed | Denied | None",
    "name": string,
    "slots": {
        // see Emplacements for details about the structure
    },
    "state": "Failed | Fulfilled | FulfillmentInProgress | InProgress |
 ReadyForFulfillment | Waiting",
    "kendraResponse": {
        // Only present when intent is KendraSearchIntent. For details, see
// https://docs.aws.amazon.com/kendra/latest/dg/API_Query.html#API_Query_ResponseSyntax
       }
}
```

Le `intent` champ est mappé à un objet avec les champs suivants :

État de confirmation

Indique si l'utilisateur a confirmé les créneaux correspondant à l'intention et si l'intention est prête à être exécutée. Les valeurs suivantes sont possibles :

`Confirmed`— L'utilisateur confirme que les valeurs des emplacements sont correctes.

`Denied`— L'utilisateur indique que les valeurs des emplacements sont incorrectes.

`None`— L'utilisateur n'a pas encore atteint le stade de confirmation.

name

Nom de l'intention.

slots

Informations sur les créneaux nécessaires pour atteindre l'objectif. Voir Emplacements pour plus de détails sur la structure.

state

Indique l'état d'exécution de l'intention. Les valeurs suivantes sont possibles :

`Failed`— Le bot n'a pas atteint son objectif.

`Fulfilled`— Le bot a terminé de remplir son intention.

`FulfillmentInProgress`— Le bot est en train de réaliser son intention.

`InProgress`— Le bot est en train de déterminer les valeurs des créneaux nécessaires pour atteindre l'objectif.

`ReadyForFulfillment`— Le bot a obtenu toutes les valeurs des créneaux correspondant à l'intention et est prêt à réaliser l'intention.

`Waiting`— Le bot attend une réponse de l'utilisateur (limité aux conversations en streaming).

kendraResponse

Contient des informations sur les résultats de la requête de recherche Kendra. Ce champ n'apparaît que si l'intention est une`KendraSearchIntent`. Consultez la syntaxe de réponse dans l'appel d'API Query pour Kendra pour plus d'informations.

Emplacements

Le `slots` champ existe au sein d'une `intent` structure et est mappé à une structure dont les clés sont les noms des emplacements prévus à cet effet. Si l'emplacement n'est pas un emplacement à valeurs multiples (voir Utilisation de plusieurs valeurs dans un emplacement pour plus de détails), il est mappé à une structure au format suivant. Notez que c'`shape`est le cas`Scalar`.

```
{
```

```
    slot name: {
        "shape": "Scalar",
        "value": {
            "originalValue": string,
            "interpretedValue": string,
            "resolvedValues": [
                string,
                ...
            ]
        }
    }
}
```

S'il s'agit d'un emplacement à valeurs multiples, l'objet auquel il est mappé contient un autre champ appelé`values`, qui est mappé à une liste de structures, chacune contenant des informations sur un emplacement constituant l'emplacement à valeurs multiples. Le format de chaque objet de la liste correspond à celui de l'objet auquel un emplacement normal est mappé. Notez que `shape` c'est le `List` cas, mais que `shape` le composant se trouve en dessous l'`values`est`Scalar`.

```
{
    slot name: {
    "shape": "List",
    "value": {
        "originalValue": string,
        "interpretedValue": string,
        "resolvedValues": [
            string,
            ...
        ]
    },
    "values": [
        {
            "shape": "Scalar",
            "value": {
                "originalValue": string,
                "interpretedValue": string,
                "resolvedValues": [
                    string,
                    ...
                ]
            }
        },
        {
```

```
            "shape": "Scalar",
            "value": {
                "originalValue": string,
                "interpretedValue": string,
                "resolvedValues": [
                    string,
                    ...
                ]
            }
        },
        ...
    ]
}
```

Les champs de l'objet slot sont décrits ci-dessous :

shape

La forme de la fente. Cette valeur correspond `List` à la présence de plusieurs valeurs dans le slot (voir Utilisation de plusieurs valeurs dans un emplacement pour plus de détails), mais c'est le `Scalar` cas contraire.

value

Un objet contenant des informations sur la valeur fournie par l'utilisateur pour un emplacement et sur l'interprétation d'Amazon Lex, au format suivant :

```
{
    "originalValue": string,
    "interpretedValue": string,
    "resolvedValues": [
        string,
        ...
    ]
}
```

Les champs sont décrits ci-dessous :

- OriginalValue — La partie de la réponse de l'utilisateur à la demande d'emplacement déterminée par Amazon Lex est liée à la valeur de l'emplacement.
- InterpretedValue — La valeur qu'Amazon Lex détermine pour le slot, en fonction des données saisies par l'utilisateur.

- ResolvedValues : liste de valeurs définies par Amazon Lex comme des résolutions possibles pour les données saisies par l'utilisateur.

values

Liste d'objets contenant des informations sur les emplacements qui constituent l'emplacement à valeurs multiples. Le format de chaque objet correspond à celui d'un emplacement normal, avec les `value` champs `shape` et décrits ci-dessus. `values`n'apparaît que si le slot est composé de plusieurs valeurs (voir Utilisation de plusieurs valeurs dans un emplacement pour plus de détails). L'objet JSON suivant montre deux emplacements de composants :

```
"values": [
    {
        "shape": "Scalar",
        "value": {
            "originalValue": string,
            "interpretedValue": string,
            "resolvedValues": [
                string,
                ...
            ]
        }
    },
    {
        "shape": "Scalar",
        "value": {
            "originalValue": string,
            "interpretedValue": string,
            "resolvedValues": [
                string,
                ...
            ]
        }
    },
    ...
]
```

État de la session

Le `sessionState` champ est mappé à un objet contenant des informations sur l'état de la conversation avec l'utilisateur. Les champs réels qui apparaissent dans l'objet dépendent du type

d'action de dialogue. Consultez Champs obligatoires dans la réponse les champs obligatoires dans une réponse Lambda. Le format de l'`sessionState`objet est le suivant :

```
"sessionState": {
    "activeContexts": [
        {
            "name": string,
            "contextAttributes": {
                string: string
            },
            "timeToLive": {
                "timeToLiveInSeconds": number,
                "turnsToLive": number
            }
        },
        ...
    ],
    "sessionAttributes": {
        string: string,
        ...
    },
    "runtimeHints": {
        "slotHints": {
            intent name: {
                slot name: {
                    "runtimeHintValues": [
                        {
                            "phrase": string
                        },
                        ...
                    ]
                },
                ...
            },
            ...
        }
    },
    "dialogAction": {
        "slotElicitationStyle": "Default | SpellByLetter | SpellByWord",
        "slotToElicit": string,
        "type": "Close | ConfirmIntent | Delegate | ElicitIntent | ElicitSlot"
    },
    "intent": {
```

```
        // see Intention for details about the structure
    },
    "originatingRequestId": string
}
```

Les champs sont décrits ci-dessous :

Contextes actifs

Liste d'objets contenant des informations sur un contexte utilisé par un utilisateur au cours d'une session. Utilisez des contextes pour faciliter et contrôler la reconnaissance des intentions. Pour plus d'informations sur les contextes, consultezDéfinition du contexte de l'intention. Chaque objet est formaté comme suit :

```
{
    "name": string,
    "contextAttributes": {
        string: string
    },
    "timeToLive": {
        "timeToLiveInSeconds": number,
        "turnsToLive": number
    }
}
```

Les champs sont décrits ci-dessous :

- name — Le nom du contexte.
- ContextAttributes — Objet contenant les noms des attributs du contexte et les valeurs auxquelles ils sont mappés.
- timeToLive— Objet qui indique la durée pendant laquelle le contexte reste actif. Cet objet peut contenir l'un des champs suivants ou les deux :
  - timeToLiveInSeconds— Le nombre de secondes pendant lesquelles le contexte reste actif.
  - turnsToLive— Le nombre de tours pendant lesquels le contexte reste actif.

sessionAttributes

Une carte de paires clé/valeur représentant des informations contextuelles spécifiques à la session. Pour en savoir plus, consultez Définition des attributs de session. L'objet est formaté comme suit :

```
{
    string: string,
    ...
}
```

Conseils d'exécution

Fournit des indications sur les phrases qu'un client est susceptible d'utiliser pour désigner un emplacement afin d'améliorer la reconnaissance audio. Les valeurs que vous indiquez dans les indices améliorent la reconnaissance audio de ces valeurs par rapport à des mots au son similaire. Le format de l'`runtimeHints`objet est le suivant :

```
{
    "slotHints": {
        intent name: {
            slot name: {
                "runtimeHintValues": [
                    {
                        "phrase": string
                    },
                    ...
                ]
            },
            ...
        },
        ...
    }
}
```

Le `slotHints` champ correspond à un objet dont les champs sont les noms des intentions du bot. Chaque nom d'intention correspond à un objet dont les champs sont les noms des emplacements correspondant à cette intention. Chaque nom d'emplacement correspond à une structure comportant un seul champ`runtimeHintValues`, qui est une liste d'objets. Chaque objet contient un `phrase` champ correspondant à un indice.

dialogAction

Détermine la prochaine action à exécuter par Amazon Lex V2. Le format de l'objet est le suivant :

```
{
    "slotElicitationStyle": "Default | SpellByLetter | SpellByWord",
```

```
    "slotToElicit": string,
    "type": "Close | ConfirmIntent | Delegate | ElicitIntent | ElicitSlot"
}
```

Les champs sont décrits ci-dessous :

- slotElicitationStyle— Détermine la manière dont Amazon Lex V2 interprète l'entrée audio de l'utilisateur si `dialogAction` c'est `ElicitSlot` le `type` cas. Pour en savoir plus, consultez Capture de valeurs de créneaux avec des styles d'orthographe. Les valeurs suivantes sont possibles :

  `Default`— Amazon Lex V2 interprète l'entrée audio de la manière par défaut pour remplir un créneau.

  `SpellByLetter`— Amazon Lex V2 écoute l'orthographe de la valeur de l'emplacement donnée par l'utilisateur.

  `SpellByWord`— Amazon Lex V2 écoute l'orthographe de la valeur de l'emplacement par l'utilisateur à l'aide de mots associés à chaque lettre (par exemple, « a as in apple »).
- slotToElicit— Définit l'emplacement à obtenir auprès de l'utilisateur si le nom `type` de `dialogAction` est `ElicitSlot`
- type — Définit l'action que le bot doit exécuter. Les valeurs suivantes sont possibles :

  `Delegate`— Laisse Amazon Lex V2 déterminer l'étape suivante.

  `ElicitIntent`— Invite le client à exprimer son intention.

  `ConfirmIntent`— Confirme les valeurs des créneaux du client et indique si l'intention est prête à être exécutée.

  `ElicitSlot`— Invite le client à fournir une valeur de créneau pour une intention.

  `Close`— Met fin au processus de réalisation des intentions.

intention

Voir Intention pour la structure du `intent` champ.

originatingRequestId

Identifiant unique pour la demande. Ce champ est facultatif pour la réponse Lambda.

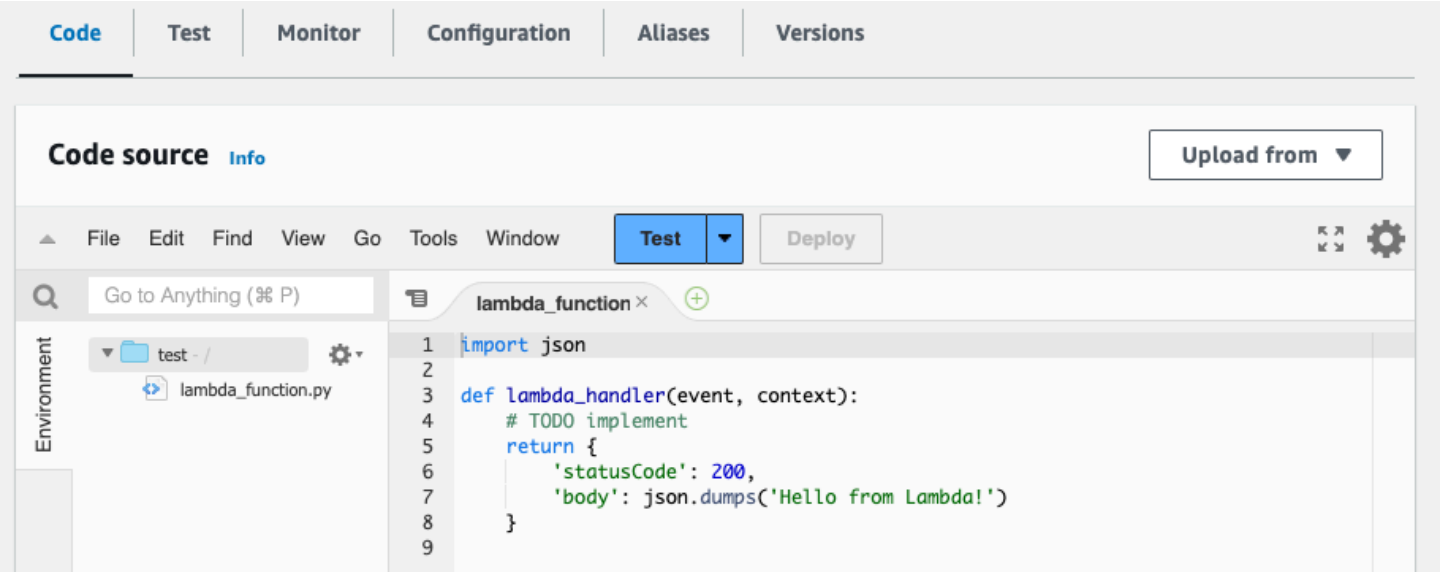
Création et association d'une fonction Lambda à un alias de bot

Création de la fonction Lambda

Pour créer une fonction Lambda pour votre bot Amazon Lex V2, accédez AWS Lambda à votre fonction AWS Management Console et créez-en une nouvelle. Vous pouvez consulter le guide du AWS Lambda développeur pour plus de détails surAWS Lambda.

1. Connectez-vous à la AWS Management Console et ouvrez la console AWS Lambda à l'adresse https://console.aws.amazon.com/lambda/.
2. Choisissez Fonctions dans la barre latérale gauche.
3. Sélectionnez Create function (Créer une fonction).
4. Vous pouvez sélectionner Auteur à partir de zéro pour commencer avec un minimum de code, utiliser un plan pour sélectionner un exemple de code pour les cas d'utilisation courants dans une liste, ou une image de conteneur pour sélectionner une image de conteneur à déployer pour votre fonction. Si vous sélectionnez Auteur à partir de zéro, poursuivez les étapes suivantes :
   a. Donnez à votre fonction un nom de fonction significatif pour décrire ce qu'elle fait.
   b. Choisissez une langue dans le menu déroulant sous Runtime pour y écrire votre fonction.
   c. Sélectionnez une architecture de jeu d'instructions pour votre fonction.
   d. Par défaut, Lambda crée un rôle avec des autorisations de base. Pour utiliser un rôle existant ou pour créer un rôle à l'aide de modèles de AWS politique, développez le menu Modifier le rôle d'exécution par défaut et sélectionnez une option.
   e. Développez le menu Paramètres avancés pour configurer d'autres options.
5. Sélectionnez Create function (Créer une fonction).

L'image suivante montre ce que vous voyez lorsque vous créez une nouvelle fonction à partir de zéro :



La fonction du gestionnaire Lambda varie en fonction de la langue que vous utilisez. Il prend au minimum un objet `event` JSON comme argument. Vous pouvez voir les champs fournis par Amazon Lex V2 à l'adresse[Interprétation du format d'événement d'entrée](). `event` Modifiez la fonction de gestion pour finalement renvoyer un objet `response` JSON correspondant au format décrit dans[Préparation du format de réponse]().

Une fois que vous avez terminé d'écrire votre fonction, sélectionnez Déployer pour autoriser l'utilisation de la fonction.

N'oubliez pas que vous pouvez associer chaque alias de bot à au plus une fonction Lambda. Cependant, vous pouvez définir autant de fonctions que nécessaire pour votre bot dans le code Lambda et appeler ces fonctions dans la fonction du gestionnaire Lambda. Par exemple, alors que toutes les intentions d'un même alias de bot doivent appeler la même fonction Lambda, vous pouvez créer une fonction de routeur qui active une fonction distincte pour chaque intention. Voici un exemple de fonction de routeur que vous pouvez utiliser ou modifier pour votre application :

```
import os
import json
import boto3

# reuse client connection as global
client = boto3.client('lambda')

def router(event):
    intent_name = event['sessionState']['intent']['name']
    fn_name = os.environ.get(intent_name)
```

```
    print(f"Intent: {intent_name} -> Lambda: {fn_name}")
    if (fn_name):
        # invoke lambda and return result
        invoke_response = client.invoke(FunctionName=fn_name, Payload =
 json.dumps(event))
        print(invoke_response)
        payload = json.load(invoke_response['Payload'])
        return payload
    raise Exception('No environment variable for intent: ' + intent_name)

def lambda_handler(event, context):
    print(event)
    response = router(event)
    return response
```

Ajouter et appeler une fonction Lambda

Pour appeler la fonction Lambda dans votre bot Amazon Lex V2, vous devez d'abord associer la fonction à un alias de bot, puis définir les points de la conversation auxquels le bot invoque la fonction. Vous pouvez effectuer ces étapes à l'aide de la console ou des opérations de l'API.

Vous pouvez utiliser les fonctions Lambda aux points suivants lors d'une conversation avec un utilisateur :

- Dans la réponse initiale, une fois l'intention reconnue. Par exemple, une fois que l'utilisateur a indiqué vouloir commander une pizza.
- Après avoir obtenu une valeur de slot auprès de l'utilisateur. Par exemple, une fois que l'utilisateur a indiqué au bot la taille de la pizza qu'il souhaite commander.
- Entre chaque nouvelle tentative pour obtenir un emplacement. Par exemple, si le client n'utilise pas une taille de pizza reconnue.
- Lors de la confirmation d'une intention. Par exemple, lors de la confirmation d'une commande de pizza.
- Pour réaliser une intention. Par exemple, pour commander une pizza.
- Une fois l'intention atteinte, et avant que votre bot ne ferme la conversation. Par exemple, pour passer à l'intention de commander un verre.

Rubriques

- Utilisation de la console

- Utilisation des opérations d'API

Utilisation de la console

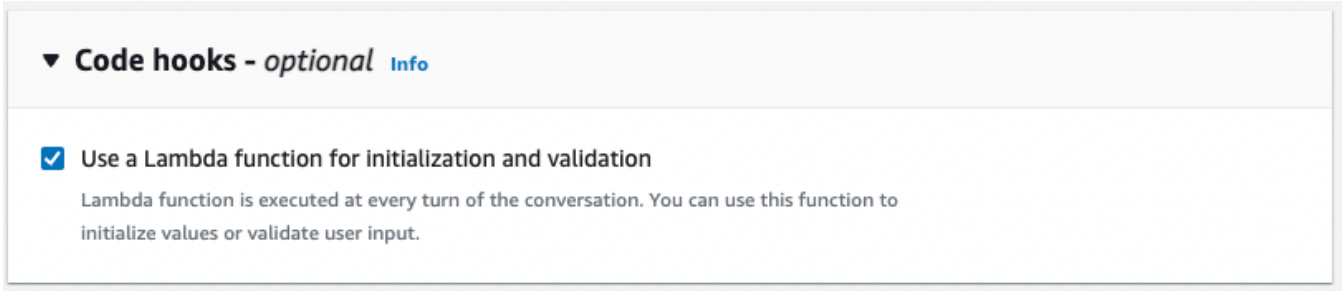
Associer une fonction Lambda à un alias de bot

1. Connectez-vous à la console Amazon Lex AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/lex/.
2. Choisissez Bots dans le panneau de gauche et dans la liste des robots, choisissez le nom du bot auquel vous souhaitez associer une fonction Lambda.
3. Dans le panneau de gauche, sélectionnez Alias dans le menu Déploiement.
4. Dans la liste des alias, choisissez le nom de l'alias auquel vous souhaitez associer une fonction Lambda.
5. Dans le panneau Langues, sélectionnez la langue dans laquelle vous souhaitez utiliser une fonction Lambda. Sélectionnez Gérer les langues dans un alias pour ajouter une langue si elle n'est pas présente dans le panneau.
6. Dans le menu déroulant Source, choisissez le nom de la fonction Lambda que vous souhaitez associer.
7. Dans le menu déroulant Version ou alias de la fonction Lambda, choisissez la version ou l'alias de la fonction Lambda que vous souhaitez utiliser. Sélectionnez ensuite Save (Enregistrer). La même fonction Lambda est utilisée à toutes fins utiles dans un langage pris en charge par le bot.

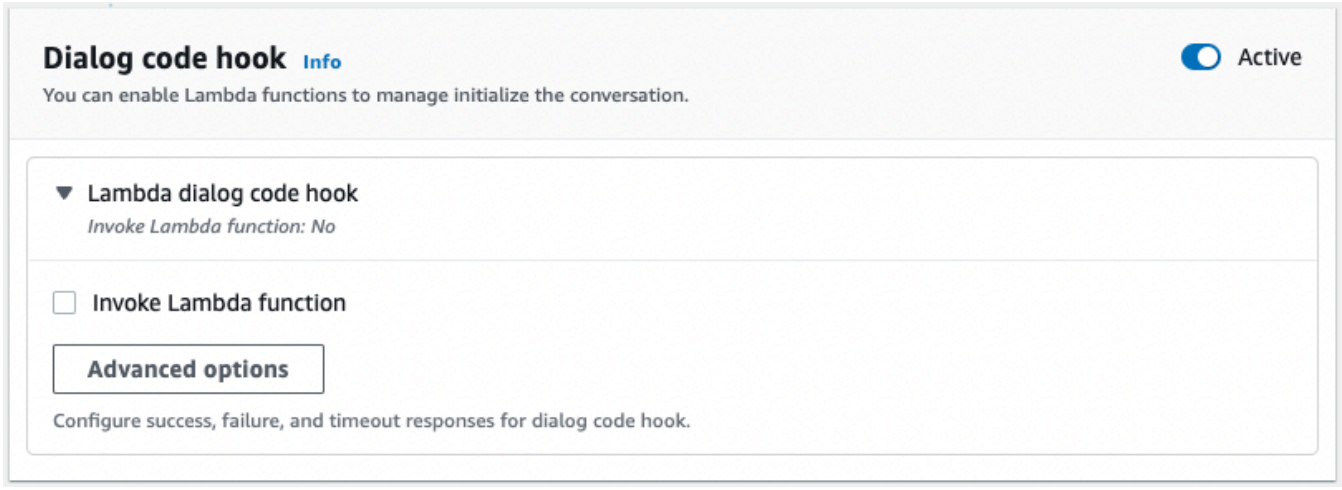
Définissez l'intention d'invoquer la fonction Lambda

1. Après avoir sélectionné un bot, sélectionnez Intents dans le menu de gauche sous la langue du bot pour lequel vous souhaitez appeler la fonction Lambda.
2. Choisissez l'intention dans laquelle vous souhaitez appeler la fonction Lambda pour ouvrir l'éditeur d'intention.
3. Il existe deux options pour configurer le crochet de code Lambda :

    1. Pour appeler la fonction Lambda après chaque étape de la conversation, accédez à la section Code hooks en bas de l'éditeur d'intention et cochez la case Utiliser une fonction Lambda pour l'initialisation et la validation, comme dans l'image suivante :



2. Vous pouvez également utiliser la section Dialog code hook dans les étapes de conversation au cours desquelles vous pouvez invoquer la fonction Lambda. La section Dialog Code Hook apparaît comme suit :



Il existe deux méthodes pour contrôler la manière dont Amazon Lex V2 appelle le code hook pour obtenir une réponse :

- Activez le bouton Actif pour le marquer comme actif ou inactif. Lorsqu'un crochet de code est actif, Amazon Lex V2 appelle le crochet de code. Lorsque le crochet de code est inactif, Amazon Lex V2 n'exécute pas le crochet de code.
- Développez la section relative au crochet de code de la boîte de dialogue Lambda et cochez la case Invoke Lambda function pour la marquer comme activée ou désactivée. Vous ne pouvez activer ou désactiver un crochet de code que lorsqu'il est marqué comme actif. Lorsqu'il est marqué comme activé, le crochet de code est exécuté normalement. Lorsqu'il est désactivé, le crochet de code n'est pas appelé et Amazon Lex V2 agit comme si le crochet de code avait été renvoyé correctement. Pour configurer les réponses une fois que le crochet de code de la boîte de dialogue a réussi, échoué ou expiré, sélectionnez Options avancées

Le crochet de code Lambda peut être invoqué aux étapes de conversation suivantes :

- Pour appeler la fonction en tant que réponse initiale, accédez à la section Réponse initiale, déployez la flèche à côté de Réponse pour accuser réception de la demande de l'utilisateur, puis sélectionnez Options avancées. Recherchez la section Dialog code hook en bas du menu qui apparaît.
- Pour appeler la fonction après avoir sélectionné un emplacement, faites défiler l'écran jusqu'à la section Machines à sous, déployez la flèche à côté de l'invite correspondant à l'emplacement, puis sélectionnez Options avancées. Recherchez la section Dialog code hook en bas du menu qui apparaît, juste au-dessus des valeurs par défaut.

  Vous pouvez également invoquer la fonction après chaque élicitation. Pour ce faire, développez les informations Bot Elicits dans la section Slot invites, sélectionnez Plus d'options d'invite et cochez la case à côté de Invoke Lambda code hook après chaque sollicitation.
- Pour appeler la fonction de confirmation de l'intention, accédez à la section Confirmation, déployez la flèche à côté de Invites de confirmation de l'intention, puis sélectionnez Options avancées. Recherchez la section Dialog code hook en bas du menu qui apparaît.
- Pour appeler la fonction d'exécution des intentions, faites défiler la page jusqu'à la section Exécution. Activez le bouton Active pour activer le crochet de code. Déplacez la flèche à côté de En cas d'expédition réussie, puis sélectionnez Options avancées. Cochez la case à côté de Utiliser une fonction Lambda pour l'expédition dans la section Fulfillment Lambda code hook pour activer le crochet de code.

4. Une fois que vous avez défini les étapes de conversation auxquelles vous souhaitez invoquer la fonction Lambda, créez à nouveau le bot pour tester la fonction.

Utilisation des opérations d'API

Associer une fonction Lambda à un alias de bot

Si vous créez un nouvel alias de bot, utilisez l'CreateBotAliasopération pour associer une fonction Lambda. Pour associer une fonction Lambda à un alias de bot existant, utilisez l'UpdateBotAliasopération. Modifiez le `botAliasLocaleSettings` champ pour qu'il contienne les paramètres appropriés :

```
{
    "botAliasLocaleSettings" : {
        locale: {
            "codeHookSpecification": {
```

```
                "lambdaCodeHook": {
                    "codeHookInterfaceVersion": "1.0",
                    "lambdaARN": "arn:aws:lambda:region:account-id:function:function-
name"
                }
            },
            "enabled": true
        },
        ...
    }
}
```

1. Le `botAliasLocaleSettings` champ correspond à un objet dont les clés sont les paramètres régionaux auxquels vous souhaitez associer la fonction Lambda. Consultez Langues et paramètres régionaux pris en charge la liste des paramètres régionaux pris en charge et les codes des clés valides.
2. Pour trouver la fonction `lambdaARN` for a Lambda, ouvrez la AWS Lambda console à l'adresse https://console.aws.amazon.com/lambda/home, sélectionnez Functions dans la barre latérale gauche, puis sélectionnez la fonction à associer à l'alias du bot. Sur le côté droit de la vue d'ensemble des fonctions, trouvez le `lambdaARN` sous-onglet « Function ARN ». Il doit contenir une région, un identifiant de compte et le nom de la fonction.
3. Pour permettre à Amazon Lex V2 d'appeler la fonction Lambda pour l'alias, définissez le `enabled` champ sur. `true`

Définissez l'intention d'invoquer la fonction Lambda

Pour configurer l'invocation de la fonction Lambda lors d'une intention, utilisez l'CreateIntentopération si vous créez une nouvelle intention ou l'UpdateIntentopération si vous appelez la fonction dans une intention existante. Les champs qui contrôlent l'invocation de la fonction Lambda dans les opérations d'intention sont`dialogCodeHook`,`initialResponseSetting`, `intentConfirmationSetting` et. `fulfillmentCodeHook`

Si vous invoquez la fonction pendant l'élicitation d'un emplacement, utilisez l'CreateSlotopération si vous créez un nouvel emplacement ou l'UpdateSlotopération pour appeler la fonction dans un emplacement existant. Le champ qui contrôle l'invocation de la fonction Lambda dans les opérations de slot est celui `slotCaptureSetting` de l'objet. `valueElicitationSetting`

1. Pour configurer le crochet de code de la boîte de dialogue Lambda pour qu'il s'exécute après chaque tour de conversation, définissez le `enabled` champ de l'DialogCodeHookSettingsobjet suivant dans le `dialogCodeHook` champ comme suit : `true`

```
"dialogCodeHook": {
    "enabled": boolean
}
```

2. Vous pouvez également configurer le crochet de code de la boîte de dialogue Lambda pour qu'il ne s'exécute qu'à des moments spécifiques des conversations en modifiant le `elicitationCodeHook` champ `codeHook` et/ou dans les structures correspondant aux étapes de conversation auxquelles vous souhaitez appeler la fonction. Pour utiliser le crochet de code de la boîte de dialogue Lambda pour répondre à une intention, utilisez le `fulfillmentCodeHook` champ dans l'opération CreateIntentou UpdateIntent. Les structures et les utilisations de ces trois types de code hooks sont les suivantes :

Crochet à code

Le `codeHook` champ définit les paramètres du crochet de code à exécuter à un stade donné de la conversation. Il s'agit d'un DialogCodeHookInvocationSettingobjet dont la structure est la suivante :

```
"codeHook": {
    "active": boolean,
    "enableCodeHookInvocation": boolean,
    "invocationLabel": string,
    "postCodeHookSpecification": PostDialogCodeHookInvocationSpecification object,
}
```

- Modifiez le `active` champ `true` pour qu'Amazon Lex V2 appelle le code hook à ce stade de la conversation.
- Modifiez le `enableCodeHookInvocation` champ `true` pour Amazon Lex V2 afin de permettre au crochet de code de s'exécuter normalement. Si vous le marquez`false`, Amazon Lex V2 agit comme si le code hook avait été renvoyé avec succès.
- `invocationLabel`Indique l'étape de dialogue à partir de laquelle le crochet de code est invoqué.
- Utilisez le `postCodeHookSpecification` champ pour spécifier les actions et les messages qui se produisent après la réussite, l'échec ou l'expiration du délai d'expiration du crochet de code.

elicitationCodeHook

Le `elicitationCodeHook` champ définit les paramètres du crochet de code à exécuter dans le cas où un ou plusieurs emplacements doivent être réactivés. Ce scénario peut se produire si l'obtention d'un créneau échoue ou si la confirmation de l'intention est refusée. Le `elicitationCodeHook` champ est un ElicitationCodeHookInvocationSettingobjet dont la structure est la suivante :

```
"elicitationCodeHook": {
    "enableCodeHookInvocation": boolean,
    "invocationLabel": string
}
```

- Modifiez le `enableCodeHookInvocation` champ `true` pour Amazon Lex V2 afin de permettre au crochet de code de s'exécuter normalement. Si vous le marquez`false`, Amazon Lex V2 agit comme si le code hook avait été renvoyé avec succès.
- `invocationLabel`Indique l'étape de dialogue à partir de laquelle le crochet de code est invoqué.

fulfillmentCodeHook

Le `fulfillmentCodeHook` champ définit les paramètres du crochet de code à exécuter pour répondre à l'intention. Il correspond à l'FulfillmentCodeHookSettingsobjet suivant :

```
"fulfillmentCodeHook": {
    "active": boolean,
    "enabled": boolean,
    "fulfillmentUpdatesSpecification": FulfillmentUpdatesSpecification object,
    "postFulfillmentStatusSpecification": PostFulfillmentStatusSpecification object
}
```

- Modifiez le `active` champ `true` pour qu'Amazon Lex V2 appelle le code hook à ce stade de la conversation.
- Modifiez le `enabled` champ `true` pour Amazon Lex V2 afin de permettre au crochet de code de s'exécuter normalement. Si vous le marquez`false`, Amazon Lex V2 agit comme si le code hook avait été renvoyé avec succès.
- Utilisez le `fulfillmentUpdatesSpecification` champ pour spécifier les messages qui semblent informer l'utilisateur lors de la réalisation de l'intention et le moment qui leur est associé.

- Utilisez le `postFulfillmentStatusSpecification` champ pour spécifier les messages et les actions qui se produisent après la réussite, l'échec ou l'expiration du délai d'expiration du crochet de code.

Vous pouvez invoquer le crochet de code Lambda aux points suivants d'une conversation en définissant les `enabled` champs `active` et`enableCodeHookInvocation`/sur : `true`

Au cours de la réponse initiale

Pour appeler la fonction Lambda dans la réponse initiale une fois l'intention reconnue, utilisez la `codeHook` structure dans le `initialResponse` champ de l'opération CreateIntentor UpdateIntent. Le `initialResponse` champ correspond à l'InitialResponseSettingobjet suivant :

```
"initialResponse": {
    "codeHook": {
        "active": boolean,
        "enableCodeHookInvocation": boolean,
        "invocationLabel": string,
        "postCodeHookSpecification": PostDialogCodeHookInvocationSpecification object,
    },
    "initialResponse": FulfillmentUpdatesSpecification object,
    "nextStep": PostFulfillmentStatusSpecification object,
    "conditional": ConditionalSpecification object
}
```

Après l'obtention des créneaux ou pendant la réélection des créneaux

Pour appeler la fonction Lambda après avoir obtenu une valeur d'emplacement, utilisez le `slotCaptureSetting` champ situé dans le `valueElicitation` champ de l'opération or. CreateSlotUpdateSlot Le `slotCaptureSetting` champ correspond à l'SlotCaptureSettingobjet suivant :

```
"slotCaptureSetting": {
    "captureConditional": ConditionalSpecification object,
    "captureNextStep": DialogState object,
    "captureResponse": ResponseSpecification object,
    "codeHook": {
        "active": true,
        "enableCodeHookInvocation": true,
        "invocationLabel": string,
        "postCodeHookSpecification": PostDialogCodeHookInvocationSpecification object,
```

```
    },
    "elicitationCodeHook": {
        "enableCodeHookInvocation": boolean,
        "invocationLabel": string
    },
    "failureConditional": ConditionalSpecification object,
    "failureNextStep": DialogState object,
    "failureResponse": ResponseSpecification object
}
```

- Pour appeler la fonction Lambda une fois que l'élicitation des emplacements est réussie, utilisez le champ. `codeHook`
- Pour appeler la fonction Lambda après l'échec de l'élicitation des emplacements et qu'Amazon Lex V2 tente à nouveau de les activer, utilisez le champ. `elicitationCodeHook`

Après confirmation ou refus de l'intention

Pour appeler la fonction Lambda lors de la confirmation d'une intention, utilisez le `intentConfirmationSetting` champ de l'opération CreateIntentor UpdateIntent. Le `intentConfirmation` champ correspond à l'IntentConfirmationSettingobjet suivant :

```
"intentConfirmationSetting": {
    "active": boolean,
    "codeHook": {
        "active": boolean,
        "enableCodeHookInvocation": boolean,
        "invocationLabel": string,
        "postCodeHookSpecification": PostDialogCodeHookInvocationSpecification object,
    },
    "confirmationConditional": ConditionalSpecification object,
    "confirmationNextStep": DialogState object,
    "confirmationResponse": ResponseSpecification object,
    "declinationConditional": ConditionalSpecification object,
    "declinationNextStep": FulfillmentUpdatesSpecification object,
    "declinationResponse": PostFulfillmentStatusSpecification object,
    "elicitationCodeHook": {
        "enableCodeHookInvocation": boolean,
        "invocationLabel": string,
    },
    "failureConditional": ConditionalSpecification object,
    "failureNextStep": DialogState object,
```

```
    "failureResponse": ResponseSpecification object,
    "promptSpecification": PromptSpecification object
}
```

- Pour appeler la fonction Lambda une fois que l'utilisateur a confirmé l'intention et ses emplacements, utilisez le `codeHook` champ.
- Pour appeler la fonction Lambda une fois que l'utilisateur a refusé la confirmation de l'intention et qu'Amazon Lex V2 a tenté à nouveau d'obtenir des emplacements, utilisez le champ. `elicitationCodeHook`

Pendant la réalisation de l'intention

Pour appeler la fonction Lambda afin de répondre à une intention, utilisez le `fulfillmentCodeHook` champ dans l'opération CreateIntentor UpdateIntent. Le `fulfillmentCodeHook` champ correspond à l'FulfillmentCodeHookSettingsobjet suivant :

```
{
    "active": boolean,
    "enabled": boolean,
    "fulfillmentUpdatesSpecification": FulfillmentUpdatesSpecification object,
    "postFulfillmentStatusSpecification": PostFulfillmentStatusSpecification object
}
```

3. Une fois que vous avez défini les étapes de conversation auxquelles vous devez invoquer la fonction Lambda, utilisez l'`BuildBotLocale`opération pour reconstruire le bot afin de tester la fonction.

Débogage de la fonction Lambda

Amazon CloudWatch Logs est un outil de suivi des appels d'API et des métriques que vous pouvez utiliser pour déboguer vos fonctions Lambda. Lorsque vous testez votre bot dans la console ou à l'aide d'appels d'API, il CloudWatch enregistre chaque étape de la conversation. Si vous utilisez une fonction d'impression dans votre code Lambda, CloudWatch affichez-la également.

Pour afficher les CloudWatch journaux de votre fonction Lambda

1. Connectez-vous à la CloudWatch console AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/cloudwatch/.

2. Dans le menu Journaux dans la barre latérale gauche, sélectionnez Groupes de journaux.
3. Sélectionnez votre groupe de journaux de fonctions Lambda, qui doit être au format suivant. `/aws/lambda/`*`function-name`*
4. La liste des flux de journaux contient un journal pour chaque session avec un bot. Choisissez un flux de journal pour le consulter.
5. Dans la liste des événements du journal, sélectionnez la flèche droite à côté de l'horodatage pour développer les détails de cet événement. Tout ce que vous imprimez à partir de votre code Lambda apparaîtra sous forme d'événement de journal. Utilisez ces informations pour déboguer votre code.
6. Après avoir débogué votre code, pensez à déployer la fonction Lambda et, si vous utilisez la console, à recharger la fenêtre de test avant de tester à nouveau le comportement du bot.

Personnalisation des interactions avec les robots

Découvrez les fonctionnalités suivantes que vous pouvez utiliser pour personnaliser les interactions des robots avec vos utilisateurs en développant et en ajustant leur comportement par défaut :

Rubriques

Analyse du sentiment suscité par les déclarations des utilisateurs

Vous pouvez utiliser l'analyse de sentiment pour déterminer les sentiments exprimés dans un énoncé d'utilisateur. Les informations de sentiment vous permettent de gérer le flux de conversation ou d'effectuer une analyse après appel. Par exemple, si le sentiment de l'utilisateur est négatif, vous pouvez créer un flux pour transmettre une conversation à un agent humain.

Amazon Lex s'intègre à Amazon Comprehend pour détecter le sentiment des utilisateurs. La réponse d'Amazon Comprehend indique si le sentiment général à l'égard du texte est positif, neutre, négatif ou mitigé. La réponse contient le sentiment le plus probable pour l'énoncé de l'utilisateur et les scores des différentes catégories de sentiment. Le score représente la probabilité que le sentiment ait été correctement détecté.

Vous activez l'analyse des sentiments pour un bot à l'aide de la console ou de l'API Amazon Lex. Vous activez l'analyse des sentiments sur un alias pour le bot. Sur la console Amazon Lex :

1. Choisissez un alias.
2. Dans Détails, choisissez Modifier.
3. Choisissez Activer ou désactiver l'analyse des sentiments.
4. Choisissez ensuite Confirm (Confirmer) pour enregistrer vos modifications.

Si vous utilisez l'API, appelez l'opération CreateBotAlias avec le champ `detectSentiment` défini sur `true`.

Lorsque l'analyse des sentiments est activée, la réponse des RecognizeUtteranceopérations RecognizeTextand renvoie un champ appelé `sentimentResponse` dans la `interpretations`

structure avec d'autres métadonnées. Le champ `sentimentResponse` comporte deux champs, `sentiment` et `sentimentScore`, qui contiennent le résultat de l'analyse de sentiment. Si vous utilisez une fonction Lambda, le `sentimentResponse` champ est inclus dans les données d'événement envoyées à votre fonction.

Voici un exemple du champ `sentimentResponse` retourné dans le cadre de la réponse `RecognizeText` ou `RecognizeUtterance`.

```
sentimentResponse {
    "sentimentScore": {
        "mixed": 0.030585512690246105,
        "positive": 0.94992071056365967,
        "neutral": 0.0141543131828308,
        "negative": 0.00893945890665054
    },
    "sentiment": "POSITIVE"
}
```

Amazon Lex appelle Amazon Comprehend en votre nom afin de déterminer le sentiment qui sous-tend chaque énoncé traité par le bot. En activant l'analyse des sentiments, vous acceptez les conditions et les accords de service d'Amazon Comprehend. Pour plus d'informations sur la tarification d'Amazon Comprehend, consultez la page de tarification d'Amazon Comprehend.

Pour plus d'informations sur le fonctionnement de l'analyse des sentiments d'Amazon Comprehend, consultez Déterminer le sentiment dans le manuel Amazon Comprehend Developer Guide.

Utilisation des scores de confiance

Amazon Lex V2 utilise deux étapes pour déterminer ce que dit un utilisateur. La première, la reconnaissance vocale automatique (ASR), crée une transcription de l'énoncé audio de l'utilisateur. La seconde, la compréhension du langage naturel (NLU), détermine le sens de l'énoncé de l'utilisateur afin de reconnaître l'intention de l'utilisateur ou la valeur des créneaux.

Par défaut, Amazon Lex V2 renvoie le résultat le plus probable de l'ASR et du NLU. Il peut parfois être difficile pour Amazon Lex V2 de déterminer le résultat le plus probable. Dans ce cas, elle renvoie plusieurs résultats possibles ainsi qu'un score de confiance qui indique la probabilité que le résultat soit correct. Un score de confiance est une note attribuée par Amazon Lex V2 qui indique le niveau de confiance relatif dont dispose Amazon Lex dans le résultat. Les scores de confiance vont de 0,0 à 1,0.

Vous pouvez utiliser vos connaissances du domaine avec le score de confiance pour déterminer l'interprétation correcte du résultat de l'ASR ou du NLU.

Le score de confiance ASR, ou transcription, est une évaluation du degré de certitude d'Amazon Lex V2 quant à l'exactitude d'une transcription donnée. Le score de confiance NLU, ou intention, est une évaluation du degré de confiance d'Amazon Lex V2 quant à l'exactitude de l'intention spécifiée par la transcription la plus élevée. Utilisez le score de confiance qui correspond le mieux à votre application.

Rubriques

Utilisation des scores de confiance en matière d'intention

Lorsqu'un utilisateur fait un énoncé, Amazon Lex V2 utilise la compréhension du langage naturel (NLU) pour comprendre la demande de l'utilisateur et renvoyer l'intention correcte. Par défaut, Amazon Lex V2 renvoie l'intention la plus probable définie par votre bot.

Dans certains cas, il peut être difficile pour Amazon Lex V2 de déterminer l'intention la plus probable. Par exemple, l'utilisateur peut faire un énoncé ambigu, ou il peut y avoir deux intentions similaires. Pour vous aider à déterminer l'intention appropriée, vous pouvez combiner vos connaissances du domaine avec les scores de confiance de la NLU dans une liste d'interprétations. Un score de confiance est une note attribuée par Amazon Lex V2 qui indique dans quelle mesure il est certain qu'une intention est la bonne.

Pour déterminer la différence entre deux intentions au sein d'une interprétation, vous pouvez comparer leurs scores de confiance. Par exemple, si une intention a un score de confiance de 0,95 et une autre a un score de 0,65, la première intention est probablement correcte. Toutefois, si une intention a un score de 0,75 et une autre un score de 0,72, il existe une ambiguïté entre les deux intentions que vous pourriez être en mesure de différencier en utilisant les connaissances du domaine dans votre application.

Vous pouvez également utiliser les scores de confiance pour créer des applications de test qui déterminent si les modifications apportées aux énoncés d'une intention influent sur le comportement du bot. Par exemple, vous pouvez obtenir les scores de confiance des intentions d'un robot à l'aide d'un ensemble d'énoncés, puis mettre à jour les intentions avec de nouveaux énoncés. Vous pouvez ensuite vérifier les scores de confiance pour voir s'il y a eu une amélioration.

Les scores de confiance renvoyés par Amazon Lex V2 sont des valeurs comparatives. Vous ne devez pas les considérer comme un score absolu. Les valeurs peuvent changer en fonction des améliorations apportées à Amazon Lex V2.

Amazon Lex V2 renvoie l'intention la plus probable et jusqu'à 4 intentions alternatives avec leurs scores associés dans la `interpretations` structure de chaque réponse. Le code JSON suivant montre la `interpretations` structure de la réponse issue de l'RecognizeTextopération :

```
"interpretations": [
   {
      "intent": {
         "confirmationState": "string",
         "name": "string",
         "slots": {
            "string" : {
               "value": {
                  "interpretedValue": "string",
                  "originalValue": "string",
                  "resolvedValues": [ "string" ]
               }
            }
         },
         "state": "string"
      },
      "nluConfidence": number
   }
]
```

AMAZON. FallbackIntent

Amazon Lex V2 revient `AMAZON.FallbackIntent` en tête de liste dans deux situations :

1. Si les scores de confiance de toutes les intentions possibles sont inférieurs au seuil de confiance. Vous pouvez utiliser le seuil par défaut ou définir votre propre seuil. Si vous l'avez `AMAZON.KendraSearchIntent` configuré, Amazon Lex V2 le renvoie également dans ce cas.
2. Si le niveau de confiance d'interprétation pour `AMAZON.FallbackIntent` est supérieur au niveau de confiance d'interprétation de toutes les autres intentions.

Notez qu'Amazon Lex V2 n'affiche pas de score de confiance pour`AMAZON.FallbackIntent`.

Définition et modification du seuil de confiance

Le seuil de confiance doit être un nombre compris entre 0,00 et 1,00. Vous pouvez définir le seuil pour chaque langue de votre bot de la manière suivante :

Utilisation de la console Amazon Lex V2

- Pour définir le seuil lorsque vous ajoutez une langue à votre bot à l'aide de la commande Ajouter une langue, vous pouvez insérer la valeur souhaitée dans le panneau du seuil de confiance.
- Pour mettre à jour le seuil, vous pouvez sélectionner Modifier dans le panneau Détails de la langue dans la langue de votre bot. Insérez ensuite la valeur souhaitée dans le panneau du seuil de confiance.

Utilisation des opérations d'API

- Pour définir le seuil, définissez le `nluIntentConfidenceThreshold` paramètre de l'CreateBotLocaleopération.
- Pour mettre à jour le seuil de confiance, définissez le `nluIntentConfidenceThreshold` paramètre de l'UpdateBotLocaleopération.

Gestion des sessions

Pour modifier l'intention qu'Amazon Lex V2 utilise dans une conversation avec l'utilisateur, vous pouvez utiliser la réponse de la fonction Lambda de votre crochet de dialogue ou vous pouvez utiliser les API de gestion de session dans votre application personnalisée.

Utilisation d'une fonction Lambda

Lorsque vous utilisez une fonction Lambda, Amazon Lex V2 l'appelle avec une structure JSON qui contient l'entrée de la fonction. La structure JSON contient un champ appelé `currentIntent` qui contient l'intention qu'Amazon Lex V2 a identifiée comme étant l'intention la plus probable de l'énoncé de l'utilisateur. La structure JSON inclut également un `alternativeIntents` champ qui contient jusqu'à quatre intentions supplémentaires susceptibles de satisfaire l'intention de l'utilisateur. Chaque intention inclut un champ intitulé `nluIntentConfidenceScore` qui contient le score de confiance qu'Amazon Lex V2 a attribué à l'intention.

Pour utiliser une autre intention, vous devez la spécifier dans l'action `ConfirmIntent` ou dans la `ElicitSlot` boîte de dialogue de votre fonction Lambda.

Pour plus d'informations, veuillez consulter Activation d'une logique personnalisée avec des AWS Lambda fonctions.

Utilisation de l'API de gestion de session

Pour utiliser une intention différente de l'intention actuelle, utilisez l'PutSessionopération. Par exemple, si vous décidez que la première alternative est préférable à l'intention choisie par Amazon Lex V2, vous pouvez utiliser cette `PutSession` opération pour modifier les intentions afin que l'intention suivante avec laquelle l'utilisateur interagit soit celle que vous avez sélectionnée.

Pour plus d'informations, veuillez consulter Gestion des sessions avec l'API Amazon Lex V2.

Utilisation des scores de fiabilité de la transcription vocale

Lorsqu'un utilisateur émet un énoncé vocal, Amazon Lex V2 utilise la reconnaissance vocale automatique (ASR) pour transcrire la demande de l'utilisateur avant qu'elle ne soit interprétée. Par défaut, Amazon Lex V2 utilise la transcription la plus probable de l'audio à des fins d'interprétation.

Dans certains cas, il peut y avoir plusieurs transcriptions possibles de l'audio. Par exemple, un utilisateur peut émettre un énoncé avec un son ambigu, tel que « Je m'appelle John », qui peut être compris comme « Je m'appelle Juan ». Dans ce cas, vous pouvez utiliser des techniques de désambiguïsation ou combiner vos connaissances du domaine avec le score de confiance de la transcription pour déterminer quelle transcription d'une liste de transcriptions est la bonne.

Amazon Lex V2 inclut la transcription supérieure et jusqu'à deux transcriptions alternatives pour la saisie par l'utilisateur dans la demande adressée à votre fonction de crochet de code Lambda. Chaque transcription contient un score de confiance indiquant qu'il s'agit de la bonne transcription. Chaque transcription inclut également toutes les valeurs de créneau déduites de l'entrée de l'utilisateur.

Vous pouvez comparer les scores de confiance de deux transcriptions pour déterminer s'il existe une ambiguïté entre elles. Par exemple, si une transcription a un score de confiance de 0,95 et l'autre un score de confiance de 0,65, la première transcription est probablement correcte et l'ambiguïté entre les deux est faible. Si les deux transcriptions ont des scores de confiance de 0,75 et 0,72, l'ambiguïté entre elles est élevée. Vous pourrez peut-être les différencier en utilisant vos connaissances du domaine.

Par exemple, si les valeurs de créneau déduites dans deux transcriptions avec un score de confiance de 0,75 et 0,72 sont « John » et « Juan », vous pouvez demander aux utilisateurs de votre base de données l'existence de ces noms et éliminer l'une des transcriptions. Si « John » n'est pas un

utilisateur dans votre base de données et que « Juan » l'est, vous pouvez utiliser le crochet de dialogue pour remplacer la valeur du slot inféré pour le prénom par « Juan ».

Les scores de confiance renvoyés par Amazon Lex V2 sont des valeurs comparatives. Ne vous fiez pas à eux comme à un score absolu. Les valeurs peuvent changer en fonction des améliorations apportées à Amazon Lex V2.

Les scores de fiabilité de la transcription audio ne sont disponibles qu'en anglais (GB) (en_GB) et en anglais (US) (en_US). Les scores de confiance ne sont pris en charge que pour une entrée audio de 8 kHz. Les scores de fiabilité de la transcription ne sont pas fournis pour les entrées audio provenant de la fenêtre de test sur la console Amazon Lex V2, car celle-ci utilise une entrée audio 16 kHz.

> **Note**
>
> Avant de pouvoir utiliser les scores de fiabilité de la transcription audio avec un robot existant, vous devez d'abord le reconstruire. Les versions existantes d'un bot ne prennent pas en charge les scores de confiance de transcription. Vous devez créer une nouvelle version du bot pour les utiliser.

Vous pouvez utiliser les scores de confiance pour plusieurs modèles de conception de conversation :

- Si le score de confiance le plus élevé tombe en dessous d'un seuil en raison d'un environnement bruyant ou d'une mauvaise qualité du signal, vous pouvez demander à l'utilisateur de lui poser la même question pour qu'il capture un son de meilleure qualité.
- Si plusieurs transcriptions présentent des scores de confiance similaires pour les valeurs de créneau, telles que « John » et « Juan », vous pouvez comparer les valeurs avec une base de données préexistante pour éliminer les entrées, ou vous pouvez demander à l'utilisateur de sélectionner l'une des deux valeurs. Par exemple, « dites 1 pour Jean ou 2 pour Juan ».
- Si votre logique métier nécessite un changement d'intention en fonction de mots clés spécifiques dans une transcription alternative dont le score de confiance est proche de celui de la transcription supérieure, vous pouvez modifier l'intention à l'aide de la fonction Lambda de votre code hook de dialogue ou à l'aide d'opérations de gestion de session. Pour en savoir plus, consultez Gestion de session.

Amazon Lex V2 envoie la structure JSON suivante avec jusqu'à trois transcriptions pour les entrées de l'utilisateur dans votre fonction de crochet de code Lambda :

```
"transcriptions": [
    {
        "transcription": "string",
        "rawTranscription": "string",
        "transcriptionConfidence": "number",
        },
        "resolvedContext": {
            "intent": "string"
        },
        "resolvedSlots": {
            "string": {
                "shape": "List",
                "value": {
                    "originalValue": "string",
                    "resolvedValues": [
                        "string"
                    ]
                },
                "values": [
                    {
                        "shape": "Scalar",
                        "value": {
                            "originalValue": "string",
                            "resolvedValues": [
                                "string"
                            ]
                        }
                    },
                    {
                        "shape": "Scalar",
                        "value": {
                            "originalValue": "string",
                            "resolvedValues": [
                                "string"
                            ]
                        }
                    }
                ]
            }
        }
    }
]
```

La structure JSON contient le texte de transcription, l'intention résolue pour l'énoncé et les valeurs de tous les intervalles détectés dans l'énoncé. Pour la saisie de texte par l'utilisateur, les transcriptions contiennent une seule transcription avec un score de confiance de 1,0.

Le contenu des transcriptions dépend de la tournure de la conversation et de l'intention reconnue.

Pour le premier tour, l'obtention d'une intention, Amazon Lex V2 détermine les trois meilleures transcriptions. Pour la transcription supérieure, elle renvoie l'intention et toutes les valeurs de créneau déduites dans la transcription.

Lors des tours suivants, lors de l'élicitation des créneaux, les résultats dépendent de l'intention déduite pour chacune des transcriptions, comme suit.

- Si l'intention déduite de la transcription supérieure est la même que celle du tour précédent et que toutes les autres transcriptions ont la même intention, alors
  - Toutes les transcriptions contiennent des valeurs de créneaux déduites.

- Si l'intention déduite de la transcription supérieure est différente de celle du tour précédent et que toutes les autres transcriptions ont l'intention précédente, alors
  - La transcription supérieure contient les valeurs de créneau déduites pour la nouvelle intention.
  - Les autres transcriptions ont l'intention précédente et ont déduit les valeurs des créneaux pour l'intention précédente.

- Si l'intention déduite pour la transcription supérieure est différente de celle du tour précédent, si une transcription est identique à l'intention précédente et qu'une transcription est une intention différente, alors
  - La transcription supérieure contient la nouvelle intention déduite et toutes les valeurs de créneau déduites dans l'énoncé.
  - La transcription dont l'intention déduite précédente contient des valeurs de créneaux déduites correspondant à cette intention.
  - La transcription présentant une intention différente ne comporte aucun nom d'intention inféré ni aucune valeur de créneau déduite.

- Si l'intention déduite de la transcription supérieure est différente de celle du tour précédent et que toutes les autres transcriptions ont des intentions différentes, alors
  - La transcription supérieure contient la nouvelle intention déduite et toutes les valeurs de créneau déduites dans l'énoncé.
  - Les autres transcriptions ne contiennent aucune intention déduite ni aucune valeur de créneau déduite.

- Si l'intention déduite pour les deux premières transcriptions est la même et différente de celle du tour précédent, et si la troisième transcription est une intention différente, alors
  - Les deux premières transcriptions contiennent la nouvelle intention déduite et toutes les valeurs de créneau déduites dans l'énoncé.
  - La troisième transcription n'a aucun nom d'intention ni aucune valeur de créneau résolue.

Gestion de session

Pour modifier l'intention utilisée par Amazon Lex V2 lors d'une conversation avec l'utilisateur, utilisez la réponse de la fonction Lambda de votre crochet de dialogue. Vous pouvez également utiliser les API de gestion de session dans votre application personnalisée.

Utilisation d'une fonction Lambda

Lorsque vous utilisez une fonction Lambda, Amazon Lex V2 l'appelle avec une structure JSON qui contient l'entrée de la fonction. La structure JSON contient un champ appelé `transcriptions` qui contient les transcriptions possibles déterminées par Amazon Lex V2 pour l'énoncé. Le `transcriptions` champ contient une à trois transcriptions possibles, chacune avec un score de confiance.

Pour utiliser l'intention d'une autre transcription, vous devez la spécifier dans l'action `ConfirmIntent` ou dans la `ElicitSlot` boîte de dialogue de votre fonction Lambda. Pour utiliser une valeur de slot issue d'une transcription alternative, définissez la valeur dans le `intent` champ de réponse de votre fonction Lambda. Pour en savoir plus, consultez Activation d'une logique personnalisée avec des AWS Lambda fonctions.

Exemple de code

L'exemple de code suivant est une fonction Lambda en Python qui utilise des transcriptions audio pour améliorer l'expérience de conversation de l'utilisateur.

Pour utiliser l'exemple de code, vous devez avoir :

- Un bot utilisant une seule langue, soit l'anglais (GB) (en_GB) soit l'anglais (US) (en_US).
- Une seule intention, OrderBirthStone. Assurez-vous que la fonction Use a Lambda pour l'initialisation et la validation est sélectionnée dans la section Code hooks de la définition de l'intention.
- L'intention doit comporter deux emplacements, « BirthMonth » et « Nom », tous deux de type`AMAZON.AlphaNumeric`.
- Alias avec la fonction Lambda définie. Pour en savoir plus, consultez Création et association d'une fonction Lambda à un alias de bot.

```
import time
import os
import logging

logger = logging.getLogger()
logger.setLevel(logging.DEBUG)


# --- Helpers that build all of the responses ---

def elicit_slot(session_attributes, intent_request, slots, slot_to_elicit, message):
    return {
        'sessionState': {
            'dialogAction': {
                'type': 'ElicitSlot',
                'slotToElicit': slot_to_elicit
            },
            'intent': {
                'name': intent_request['sessionState']['intent']['name'],
                'slots': slots,
                'state': 'InProgress'
            },
            'sessionAttributes': session_attributes,
            'originatingRequestId': 'e3ab4d42-fb5f-4cc3-bb78-caaf6fc7cccd'
        },
        'sessionId': intent_request['sessionId'],
        'messages': [message],
        'requestAttributes': intent_request['requestAttributes'] if 'requestAttributes'
 in intent_request else None
```

```
    }


def close(intent_request, session_attributes, fulfillment_state, message):
    intent_request['sessionState']['intent']['state'] = fulfillment_state
    return {
        'sessionState': {
            'sessionAttributes': session_attributes,
            'dialogAction': {
                'type': 'Close'
            },
            'intent': intent_request['sessionState']['intent'],
            'originatingRequestId': '3ab4d42-fb5f-4cc3-bb78-caaf6fc7cccd'
        },
        'messages': [message],
        'sessionId': intent_request['sessionId'],
        'requestAttributes': intent_request['requestAttributes'] if 'requestAttributes'
 in intent_request else None
    }


def delegate(intent_request, session_attributes):
    return {
        'sessionState': {
            'dialogAction': {
                'type': 'Delegate'
            },
            'intent': intent_request['sessionState']['intent'],
            'sessionAttributes': session_attributes,
            'originatingRequestId': 'abc'
        },
        'sessionId': intent_request['sessionId'],
        'requestAttributes': intent_request['requestAttributes'] if 'requestAttributes'
 in intent_request else None
    }


def get_session_attributes(intent_request):
    sessionState = intent_request['sessionState']
    if 'sessionAttributes' in sessionState:
        return sessionState['sessionAttributes']

    return {}
```

```
def get_slots(intent_request):
    return intent_request['sessionState']['intent']['slots']


""" --- Functions that control the behavior of the bot --- """


def order_birth_stone(intent_request):
    """
    Performs dialog management and fulfillment for ordering a birth stone.
    Beyond fulfillment, the implementation for this intent demonstrates the following:
    1) Use of N best transcriptions to re prompt user when confidence for top
 transcript is below a threshold
    2) Overrides resolved slot for birth month from a known fixed list if the top
 transcript
    is not accurate.
    """

    transcriptions = intent_request['transcriptions']

    if intent_request['invocationSource'] == 'DialogCodeHook':
        # Disambiguate if there are multiple transcriptions and the top transcription
        # confidence is below a threshold (0.8 here)
        if len(transcriptions) > 1 and transcriptions[0]['transcriptionConfidence'] <
 0.8:
            if transcriptions[0]['resolvedSlots'] is not {} and 'Name' in
 transcriptions[0]['resolvedSlots'] and \
                    transcriptions[0]['resolvedSlots']['Name'] is not None:
                return prompt_for_name(intent_request)
            elif transcriptions[0]['resolvedSlots'] is not {} and 'BirthMonth' in
 transcriptions[0]['resolvedSlots'] and \
                    transcriptions[0]['resolvedSlots']['BirthMonth'] is not None:
                return validate_month(intent_request)

    return continue_conversation(intent_request)


def prompt_for_name(intent_request):
    """
    If the confidence for the name is not high enough, re prompt the user with the
 recognized names
    so it can be confirmed.
    """
```

```
    resolved_names = []
    for transcription in intent_request['transcriptions']:
        if transcription['resolvedSlots'] is not {} and 'Name' in
 transcription['resolvedSlots'] and \
                transcription['resolvedSlots']['Name'] is not None:
            resolved_names.append(transcription['resolvedSlots']['Name']['value']
['originalValue'])
    if len(resolved_names) > 1:
        session_attributes = get_session_attributes(intent_request)
        slots = get_slots(intent_request)
        return elicit_slot(session_attributes, intent_request, slots, 'Name',
                           {'contentType': 'PlainText',
                            'content': 'Sorry, did you say your name is {} ?'.format("
 or ".join(resolved_names))})
    else:
        return continue_conversation(intent_request)


def validate_month(intent_request):
    """
    Validate month from an expected list, if not valid looks for other transcriptions
 and to see if the month
    recognized there has an expected value. If there is, replace with that and if not
 continue conversation.
    """

    expected_months = ['january', 'february', 'march']
    resolved_months = []
    for transcription in intent_request['transcriptions']:
        if transcription['resolvedSlots'] is not {} and 'BirthMonth' in
 transcription['resolvedSlots'] and \
                transcription['resolvedSlots']['BirthMonth'] is not None:
            resolved_months.append(transcription['resolvedSlots']['BirthMonth']
['value']['originalValue'])

    for resolved_month in resolved_months:
        if resolved_month in expected_months:
            intent_request['sessionState']['intent']['slots']['BirthMonth']
['resolvedValues'] = [resolved_month]
            break

    return continue_conversation(intent_request)
```

```
def continue_conversation(event):
    session_attributes = get_session_attributes(event)

    if event["invocationSource"] == "DialogCodeHook":
        return delegate(event, session_attributes)


# --- Intents ---


def dispatch(intent_request):
    """
    Called when the user specifies an intent for this bot.
    """

    logger.debug('dispatch sessionId={},
 intentName={}'.format(intent_request['sessionId'],

 intent_request['sessionState']['intent']['name']))

    intent_name = intent_request['sessionState']['intent']['name']

    # Dispatch to your bot's intent handlers
    if intent_name == 'OrderBirthStone':
        return order_birth_stone(intent_request)

    raise Exception('Intent with name ' + intent_name + ' not supported')


# --- Main handler ---


def lambda_handler(event, context):
    """
    Route the incoming request based on intent.
    The JSON body of the request is provided in the event slot.

    """
    # By default, treat the user request as coming from the America/New_York time
 zone.
    os.environ['TZ'] = 'America/New_York'
    time.tzset()
    logger.debug('event={}'.format(event))
```

```
    return dispatch(event)
```

Utilisation de l'API de gestion de session

Pour utiliser une intention différente de l'intention actuelle, utilisez l'PutSessionopération. Par exemple, si vous décidez que la première alternative est préférable à l'intention choisie par Amazon Lex V2, vous pouvez utiliser l'`PutSession`opération pour modifier les intentions. Ainsi, l'intention suivante avec laquelle l'utilisateur interagira sera celle que vous avez sélectionnée.

Vous pouvez également utiliser cette `PutSession` opération pour modifier la valeur du slot dans la `intent` structure afin d'utiliser une valeur provenant d'une autre transcription.

Pour plus d'informations, consultez Gestion des sessions avec l'API Amazon Lex V2.

Personnalisation des transcriptions vocales

Le comportement par défaut de votre bot peut parfois entraîner des transcriptions vocales inexactes. Les fonctionnalités suivantes sont disponibles pour aider votre bot à reconnaître des mots ou des noms moins courants ou faciles à confondre.

Rubriques

- Améliorer la reconnaissance vocale grâce à un vocabulaire personnalisé
- Amélioration de la reconnaissance des valeurs des créneaux grâce à des indices d'exécution
- Capture de valeurs de créneaux avec des styles d'orthographe

Améliorer la reconnaissance vocale grâce à un vocabulaire personnalisé

Vous pouvez fournir à Amazon Lex V2 plus d'informations sur la façon de traiter des conversations audio avec un bot en créant un vocabulaire personnalisé dans une langue spécifique. Un vocabulaire personnalisé est une liste de phrases spécifiques que vous souhaitez qu'Amazon Lex V2 reconnaisse dans l'entrée audio. Il s'agit généralement de noms propres ou de mots spécifiques à un domaine qu'Amazon Lex V2 ne reconnaît pas.

Supposons, par exemple, que vous disposiez d'un bot de support technique. Vous pouvez ajouter « sauvegarde » à un vocabulaire personnalisé pour aider le bot à transcrire correctement l'audio en tant que « sauvegarde », même lorsque le son ressemble à « emballer ». Un vocabulaire personnalisé peut également aider à reconnaître des mots rares dans l'audio, tels que « solvabilité » pour les services financiers ou des noms propres tels que « Cognito » ou « Monitron ».

Principes de base du vocabulaire personnalisé

- Un vocabulaire personnalisé fonctionne sur la transcription des entrées audio vers un bot. Vous devez fournir des exemples d'énoncés pour reconnaître une intention ou une valeur de créneau.
- Un vocabulaire personnalisé est propre à une langue spécifique. Vous devez configurer des vocabulaires personnalisés de manière indépendante pour chaque langue. Les vocabulaires personnalisés ne sont pris en charge que pour les langues anglaise (Royaume-Uni) et anglaise (États-Unis).
- Des vocabulaires personnalisés sont disponibles avec les intégrations de centre d'appels prises en charge par Amazon Lex V2. La fenêtre de test de la console Amazon Lex V2 prend en charge les vocabulaires personnalisés pour tous les robots Amazon Lex V2 créés à compter du 31 juillet 2022. Si vous rencontrez des problèmes avec les vocabulaires personnalisés dans la fenêtre de test, reconstruisez le bot et réessayez.

Amazon Lex V2 utilise des vocabulaires personnalisés pour définir à la fois les intentions et les créneaux. Le même fichier de vocabulaire personnalisé est utilisé pour les intentions et les créneaux. Vous pouvez désactiver de manière sélective la fonctionnalité de vocabulaire personnalisé pour un créneau lorsque vous ajoutez un type de créneau.

Susciter une intention : vous pouvez créer un vocabulaire personnalisé pour susciter une intention. Ces phrases sont utilisées pour la transcription lorsque votre bot détermine l'intention de l'utilisateur. Par exemple, si vous avez configuré l'expression « sauvegarde » dans votre vocabulaire personnalisé, Amazon Lex V2 transcrit la saisie utilisateur en « pouvez-vous sauvegarder mes photos, s'il vous plaît ? » ... même lorsque le son sonne comme « peux-tu, s'il te plaît, emballer mes photos ». Vous pouvez spécifier le degré d'amplification de chaque phrase en configurant une pondération de 0, 1, 2 ou 3. Vous pouvez également spécifier une autre représentation pour la phrase dans la sortie de synthèse vocale finale en ajoutant un `displayAs` champ.

Les phrases de vocabulaire personnalisées utilisées pour améliorer la transcription lors de la sollicitation d'intentions n'affectent pas les transcriptions lors de l'obtention de créneaux. Pour plus d'informations sur la création d'un vocabulaire personnalisé pour susciter des intentions, reportez-vous à la section. Création d'un vocabulaire personnalisé pour obtenir des intentions et des créneaux

Obtenir des créneaux personnalisés — Vous pouvez utiliser un vocabulaire personnalisé pour améliorer la reconnaissance des créneaux pour les conversations audio. Pour améliorer la capacité de votre bot Amazon Lex V2 à reconnaître les valeurs des emplacements, créez un emplacement personnalisé et ajoutez-y les valeurs des emplacements, puis choisissez Utiliser les valeurs des

emplacements comme vocabulaire personnalisé. Les exemples de valeurs d'emplacement incluent les noms de produits, les catalogues ou les noms propres. Vous ne devez pas utiliser de mots ou d'expressions courants tels que « oui » et « non » dans les vocabulaires personnalisés.

Une fois les valeurs d'emplacement ajoutées, ces valeurs sont utilisées pour améliorer la reconnaissance des emplacements lorsque le bot attend une saisie pour l'emplacement personnalisé. Ces valeurs ne sont pas utilisées pour la transcription lors de l'obtention d'une intention. Pour plus d'informations, veuillez consulter Ajouter des types de slots.

Bonnes pratiques pour créer un vocabulaire personnalisé

Obtenir une intention

- Les glossaires personnalisés fonctionnent mieux lorsqu'ils ciblent des termes ou des expressions spécifiques. N'ajoutez des mots à un vocabulaire personnalisé que s'ils ne sont pas facilement reconnus par Amazon Lex V2.
- Décidez du poids à accorder à un mot en fonction de la fréquence à laquelle le mot n'est pas reconnu dans la transcription et de sa rareté dans la saisie. Les mots difficiles à prononcer nécessitent un poids plus élevé.
- Utilisez un ensemble de tests représentatif pour déterminer si un poids est approprié. Vous pouvez collecter un ensemble de tests audio en activant l'enregistrement audio dans les journaux de conversation.
- Évitez d'utiliser des mots courts tels que « on », « it », « to », « yes », « non » dans un vocabulaire personnalisé.

Obtenir un créneau personnalisé

- Ajoutez les valeurs au type d'emplacement personnalisé que vous souhaitez voir reconnu. Ajoutez toutes les valeurs d'emplacement possibles pour le type d'emplacement personnalisé, quelle que soit la fréquence ou la rareté de la valeur d'emplacement.
- Activez cette option uniquement lorsque le type d'emplacement personnalisé contient une liste de valeurs de catalogue ou d'entités telles que des noms de produits ou des fonds communs de placement.
- Désactivez cette option si le type d'emplacement est utilisé pour capturer des phrases génériques telles que « oui », « non », « je ne sais pas », « peut-être » ou des mots génériques tels que « un », « deux », « trois ».

- Limitez le nombre de valeurs d'emplacement et de synonymes à 500 ou moins pour de meilleures performances.

Entrez des acronymes ou d'autres mots dont les lettres doivent être prononcées individuellement sous forme de lettres séparées par un point et un espace. N'utilisez pas de lettres individuelles à moins qu'elles ne fassent partie d'une phrase, telle que « J.P. Morgan » ou « A. W. S. » Vous pouvez utiliser des lettres majuscules ou minuscules pour définir un acronyme.

Création d'un vocabulaire personnalisé pour obtenir des intentions et des créneaux

Vous pouvez utiliser la console Amazon Lex V2 pour créer et gérer un vocabulaire personnalisé, ou vous pouvez utiliser les opérations de l'API Amazon Lex V2. Il existe deux manières de créer un vocabulaire personnalisé via la console :

Console

Importez du vocabulaire personnalisé dans la console :

1. Ouvrez la console Amazon Lex V2 à l'adresse https://console.aws.amazon.com/lexv2/home
2. Dans la liste des robots, choisissez celui auquel vous souhaitez ajouter le vocabulaire personnalisé.
3. Sur la page détaillée du bot, dans la section Ajouter des langues, choisissez Afficher les langues.
4. Dans la liste des langues, choisissez la langue à laquelle vous souhaitez ajouter le vocabulaire personnalisé.

Créez un nouveau vocabulaire personnalisé directement via la console :

1. Cliquez sur Créer dans la section Vocabulaire personnalisé de la page de détails de la langue. Cela ouvrira une fenêtre d'édition sans vocabulaire personnalisé.
2. Ajoutez des entrées pour la phrase DisplayAs et le poids selon vos besoins. Vous pouvez également apporter des modifications en ligne aux éléments ajoutés en mettant à jour leurs champs ou en les supprimant de la liste.
3. Cliquez sur Enregistrer. Remarque : le nouveau vocabulaire personnalisé n'est enregistré dans votre bot qu'après avoir cliqué sur Enregistrer.
4. Vous pouvez continuer à apporter des modifications en ligne sur cette page et cliquer sur Enregistrer lorsque vous avez terminé.

5. Cette page vous permet également d'importer, d'exporter et de supprimer un fichier de vocabulaire personnalisé à partir du menu déroulant en haut à droite.

API

Utilisez l'**`ListCustomVocabularyItems`**API pour afficher les entrées de vocabulaire personnalisées :

1. Utilisez cette `ListCustomVocabularyItems` opération pour afficher les entrées de vocabulaire personnalisées. Le corps de la demande ressemblera à ceci :

```
{
   "maxResults": number,
   "nextToken": "string"
}
```

2. Veuillez noter que les champs `maxResults` et `nextToken` sont facultatifs pour le corps de la demande.
3. La réponse de l'`ListCustomVocabularyItems`opération ressemble à ceci :

```
{
   "botId": "string",
   "botVersion": "string",
   "localeId": "string",
   "customVocabularyItems": [
        {
            "itemId": "string",
            "phrase": "string",
            "weight": number,
            "displayAs": "string"
        }
   ]
}
```

Utilisez l'**`BatchCreateCustomVocabularyItem`**API pour créer de nouvelles entrées de vocabulaire personnalisées :

1. Si aucun vocabulaire personnalisé n'a encore été créé pour les paramètres régionaux de votre bot, veuillez suivre les étapes pour utiliser le StartImportpour créer un vocabulaire personnalisé.

2. Une fois le vocabulaire personnalisé créé, utilisez l'`BatchCreateCustomVocabularyItem`opération pour créer de nouvelles entrées de vocabulaire personnalisées. Le corps de la demande ressemblera à ceci :

```
{
   "customVocabularyItemList": [
        {
            "phrase": "string",
            "weight": number,
            "displayAs": "string"
        }
   ]
}
```

3. Veuillez noter que les champs `weight` et `displayAs` sont facultatifs pour le corps de la demande.

4. La réponse du `BatchCreateCustomVocabularyItem` ressemblera à ceci :

```
{
    "botId": "string",
    "botVersion": "string",
    "localeId": "string",
    "errors": [
        {
            "itemId": "string",
            "errorMessage": "string",
            "errorCode": "string"
        }
    ],
    "resources": [
        {
            "itemId": "string",
            "phrase": "string",
            "weight": number,
            "displayAs": "string"
        }
    ]
}
```

5. Comme il s'agit d'une opération par lots, la demande n'échouera pas si l'un des éléments ne parvient pas à être créé. La liste des erreurs contiendra des informations sur les raisons de

l'échec de l'opération pour cette entrée spécifique. La liste des ressources contiendra toutes les entrées qui ont été créées avec succès.

6. En effetBatchCreateCustomVocabularyItem, vous pouvez vous attendre à voir les types d'erreurs suivants :

   - RESOURCE_DOES_NOT_EXIST: Le vocabulaire personnalisé n'existe pas. Suivez les étapes de création d'un vocabulaire personnalisé avant de lancer cette opération.
   - DUPLICATE_INPUT: La liste des entrées contient des phrases dupliquées.
   - RESOURCE_ALREADY_EXISTS: La phrase spécifiée pour l'entrée existe déjà dans votre vocabulaire personnalisé.
   - INTERNAL_SERVER_FAILURE: Une erreur s'est produite dans le backend lors du traitement de votre demande. Cela peut indiquer une panne de service ou un autre problème.

Utilisez l'**BatchDeleteCustomVocabularyItem**API pour supprimer des entrées de vocabulaire personnalisées existantes :

1. Si aucun vocabulaire personnalisé n'a encore été créé pour les paramètres régionaux de votre bot, veuillez suivre les étapes décrites dans la section Utiliser le StartImportpour créer un vocabulaire personnalisé pour en créer un.
2. Une fois le vocabulaire personnalisé créé, utilisez l'BatchDeleteCustomVocabularyItemopération pour supprimer les entrées de vocabulaire personnalisées existantes. Le corps de la demande ressemblera à ceci :

```
{
   "customVocabularyItemList": [
        {
            "itemId": "string"
        }
   ]
}
```

3. La réponse du BatchDeleteCustomVocabularyItem ressemblera à ceci :

```
{
    "botId": "string",
    "botVersion": "string",
    "localeId": "string",
    "errors": [
```

```
        {
            "itemId": "string",
            "errorMessage": "string",
            "errorCode": "string"
        }
    ],
    "resources": [
        {
            "itemId": "string",
            "phrase": "string",
            "weight": number,
            "displayAs": "string"
        }
    ]
}
```

4. Comme il s'agit d'une opération par lots, la demande n'échouera pas si l'un des éléments ne parvient pas à être supprimé. La liste des erreurs contiendra des informations sur les raisons de l'échec de l'opération pour cette entrée spécifique. La liste des ressources contiendra toutes les entrées qui ont été supprimées avec succès.
5. En effet`BatchDeleteCustomVocabularyItem`, vous pouvez vous attendre à voir les types d'erreurs suivants :

   - `RESOURCE_DOES_NOT_EXIST`: l'entrée de vocabulaire personnalisée que vous essayez de supprimer n'existe pas.
   - `INTERNAL_SERVER_FAILURE`: Une erreur s'est produite dans le backend lors du traitement de votre demande. Cela peut indiquer une panne de service ou un autre problème.

Utilisez l'**`BatchUpdateCustomVocabularyItem`**API pour mettre à jour les entrées de vocabulaire personnalisées existantes :

1. Si aucun vocabulaire personnalisé n'a encore été créé pour les paramètres régionaux de votre bot, veuillez suivre les étapes décrites dans la section Utiliser le StartImportpour créer un vocabulaire personnalisé.
2. Une fois le vocabulaire personnalisé créé, utilisez l'`BatchUpdateCustomVocabularyItem`opération pour mettre à jour les entrées de vocabulaire personnalisées existantes. Le corps de la demande ressemblera à ceci :

```
{
```

```
   "customVocabularyItemList": [
        {
            "itemId": "string",
            "phrase": "string",
            "weight": number,
            "displayAs": "string"
        }
   ]
}
```

3. Veuillez noter que les champs `weight` et `displayAs` sont facultatifs pour le corps de la demande.
4. La réponse du `BatchUpdateCustomVocabularyItem` ressemblera à ceci :

```
{
    "botId": "string",
    "botVersion": "string",
    "localeId": "string",
    "errors": [
        {
            "itemId": "string",
            "errorMessage": "string",
            "errorCode": "string"
        }
    ],
    "resources": [
        {
            "itemId": "string",
            "phrase": "string",
            "weight": number,
            "displayAs": "string"
        }
    ]
}
```

5. Comme il s'agit d'une opération par lots, la demande n'échouera pas si l'un des éléments ne parvient pas à être supprimé. La liste des erreurs contiendra des informations sur les raisons de l'échec de l'opération pour cette entrée spécifique. La liste des ressources contiendra toutes les entrées qui ont été mises à jour avec succès.
6. En effet`BatchUpdateCustomVocabularyItem`, vous pouvez vous attendre à voir les types d'erreurs suivants :

- `RESOURCE_DOES_NOT_EXIST`: l'entrée de vocabulaire personnalisée que vous essayez de mettre à jour n'existe pas.
- `DUPLICATE_INPUT`: La liste des entrées contient des ItemID dupliqués.
- `RESOURCE_ALREADY_EXISTS`: La phrase spécifiée pour l'entrée existe déjà dans votre vocabulaire personnalisé.
- `INTERNAL_SERVER_FAILURE`: Une erreur s'est produite dans le backend lors du traitement de votre demande. Cela peut indiquer une panne de service ou un autre problème.

Création d'un fichier de vocabulaire personnalisé

Un fichier de vocabulaire personnalisé est une liste de valeurs séparées par des tabulations qui contient la phrase à reconnaître, une pondération pour la renforcer et un `displayAs` champ qui remplacera la phrase dans la transcription du discours. Les phrases dont la valeur d'amplification est plus élevée sont plus susceptibles d'être utilisées lorsqu'elles apparaissent dans l'entrée audio.

Le fichier de vocabulaire personnalisé doit être nommé **`CustomVocabulary.tsv`** et doit être compressé dans un fichier zip avant de pouvoir être importé. La taille du fichier zip doit être inférieure à 300 Mo. Le nombre maximum de phrases dans un vocabulaire personnalisé est de 500.

- phrase de 1 à 4 mots qui doivent être reconnus. Séparez les mots de la phrase par des espaces. Vous ne pouvez pas contenir de phrases dupliquées dans le fichier. Le champ de phrase est obligatoire.
- poids — La mesure dans laquelle la reconnaissance de la phrase est améliorée. La valeur est un entier 0, 1, 2 ou 3. Si vous ne spécifiez pas de poids, la valeur par défaut est 1. Décidez de la pondération en fonction de la fréquence à laquelle le mot n'est pas reconnu dans la transcription et de la rareté du mot dans l'entrée. La pondération 0 signifie qu'aucun renforcement ne sera appliqué et que l'entrée ne sera utilisée que pour effectuer des remplacements en utilisant le `displayAs` champ.
- DisplayAs : définit l'apparence que vous souhaitez donner à votre phrase dans votre sortie de transcription. Ce champ est facultatif dans le vocabulaire personnalisé.

Le fichier de vocabulaire personnalisé doit contenir une ligne d'en-tête avec les en-têtes « phrase », « poids » et « DisplayAs ». Les en-têtes peuvent être dans n'importe quel ordre, mais ils doivent suivre la nomenclature ci-dessus.

L'exemple suivant est un fichier de vocabulaire personnalisé. Le caractère de tabulation requis pour séparer la phrase, le poids et le DisplayAs est représenté par le texte « [TAB] ». Si vous utilisez cet exemple, remplacez le texte par un caractère de tabulation.

```
phrase[TAB]weight[TAB]displayAs
Newcastle[TAB]2
Hobart[TAB]2[TAB]Hobart, Australia
U. Dub[TAB]1[TAB]University of Washington, Seattle
W. S. U.[TAB]3
Issaquah
Kennewick
```

Amélioration de la reconnaissance des valeurs des créneaux grâce à des indices d'exécution

Grâce aux conseils d'exécution, vous pouvez attribuer à Amazon Lex V2 un ensemble de valeurs de créneaux en fonction du contexte afin d'obtenir une meilleure reconnaissance dans les conversations audio et d'améliorer la résolution des créneaux. Vous pouvez utiliser les indices d'exécution pour fournir une liste de phrases au moment de l'exécution susceptibles d'être utilisées pour la résolution d'une valeur de créneau.

Par exemple, si un utilisateur interagissant avec un robot de réservation de vols se rend fréquemment à San Francisco, Jakarta, Séoul et Moscou, vous pouvez configurer des indices d'exécution avec une liste de ces quatre villes lorsque vous sélectionnez la destination afin de mieux reconnaître les villes les plus fréquentées.

Les conseils d'exécution ne sont disponibles qu'en anglais (États-Unis) et en anglais (Royaume-Uni). Ils peuvent être utilisés avec les types de machines à sous suivants :

- Types de machines à sous personnalisés
- Amazon.City
- Amazon.country
- AMAZON. FirstName
- AMAZON. LastName
- État d'Amazon
- AMAZON. StreetName

Conseils de base sur l'exécution

- Les indices d'exécution ne sont utilisés que pour obtenir une valeur d'emplacement auprès d'un utilisateur.
- Lorsque vous utilisez des indices d'exécution, leurs valeurs sont préférées aux valeurs similaires. Par exemple, pour un robot qui commande des plats, vous pouvez définir une liste d'éléments de menu à titre d'indices d'exécution tout en sélectionnant des plats dans un emplacement personnalisé pour préférer « filet » à un « gars » au son similaire.
- Si l'entrée utilisateur est différente des valeurs fournies dans les indications d'exécution, l'entrée utilisateur d'origine sera utilisée pour le slot.
- Pour les types d'emplacements personnalisés, les valeurs fournies sous forme d'indicateurs d'exécution seront utilisées pour la résolution de l'emplacement même si elles ne font pas partie de l'emplacement personnalisé lors de la création du bot.
- Les indices d'exécution ne sont pris en charge que pour une entrée audio de 8 kHz. Ils sont disponibles avec les intégrations de centres de contact prises en charge par Amazon Lex V2. Les indications d'exécution ne sont pas fournies pour l'entrée audio depuis la fenêtre de test de la console Amazon Lex V2, car celle-ci utilise une entrée audio 16 kHz.

> **Note**
>
> Avant de pouvoir utiliser les indices d'exécution avec un bot existant, vous devez d'abord le reconstruire. Les versions existantes d'un bot ne prennent pas en charge les indices d'exécution. Vous devez créer une nouvelle version du bot pour les utiliser.

Vous pouvez envoyer des indications d'exécution à Amazon Lex V2 à l'aide de l'StartConversationopération PutSessionRecognizeTextRecognizeUtterance,, ou. Vous pouvez également ajouter des indices d'exécution à l'aide d'une fonction Lambda.

Vous pouvez envoyer des conseils d'exécution au début d'une conversation afin de configurer les conseils pour chaque emplacement utilisé dans le bot, ou vous pouvez envoyer des conseils dans le cadre de l'état de la session pendant une conversation. L'`runtimeHints`attribut fait correspondre un emplacement aux indications relatives à cet emplacement.

Une fois que vous avez envoyé un indice d'exécution à Amazon Lex V2, il persiste à chaque étape de la conversation jusqu'à la fin de la session. Si vous envoyez une `runtimeHints` structure nulle, les indices existants sont utilisés. Vous pouvez modifier les indices en :

- Envoi d'une nouvelle `runtimeHints` structure au bot. Le contenu de la nouvelle structure remplace celui existant.
- Envoi d'une `runtimeHints` structure vide au bot. Cela efface les indications d'exécution pour le bot.

Ajouter des valeurs de créneau dans le contexte

Ajoutez du contexte à votre bot en fournissant les valeurs d'emplacement attendues comme indications d'exécution lorsque votre application dispose d'informations sur le prochain énoncé probable de l'utilisateur. Ajoutez un crochet de code de dialogue Lambda à votre bot (voir Activation d'une logique personnalisée avec des AWS Lambda fonctions pour plus d'informations) et utilisez le proposedNextStatechamp du Interprétation du format d'événement d'entrée pour déterminer les conseils d'exécution à inclure pour améliorer la conversation avec l'utilisateur.

Par exemple, dans une application bancaire, vous pouvez générer une liste de pseudonymes de compte pour un utilisateur spécifique, puis utiliser cette liste pour sélectionner le compte auquel l'utilisateur souhaite accéder.

Envoyez des conseils d'exécution au début de la conversation lorsque vous disposez d'un contexte pour aider votre bot à interpréter les données saisies par les utilisateurs. Par exemple, si vous avez le numéro de téléphone de l'utilisateur, vous pouvez utiliser ces informations pour rechercher l'utilisateur afin de pouvoir utiliser l'`StartConversation`opération `PutSession` ou pour transmettre des indications sur le prénom et le nom de famille au bot si vous demandez le nom de l'utilisateur pour valider ses informations d'identification.

Au cours d'une conversation, vous pouvez recueillir des informations à partir d'une valeur d'emplacement qui peuvent vous aider avec une autre valeur d'emplacement. Par exemple, dans une application d'entretien automobile, lorsque vous avez le numéro de compte de l'utilisateur, vous pouvez effectuer une recherche pour trouver les voitures que possède le client et les transmettre comme indices à un autre emplacement.

Entrez des acronymes ou d'autres mots dont les lettres doivent être prononcées individuellement, sous forme de lettres simples séparées par un point et un espace. N'utilisez pas de lettres individuelles à moins qu'elles ne fassent partie d'une phrase, telle que « J.P. Morgan » ou « A.W.S ». Vous pouvez utiliser des lettres majuscules ou minuscules pour définir un acronyme.

Ajouter des indices à un emplacement

Pour ajouter des indications d'exécution à un slot, vous devez utiliser la `runtimeHints` structure qui en fait partie. `sessionState` Voici un exemple de `runtimeHints` structure. Il fournit des indications pour deux emplacements, « FirstName » et « LastName » pour l'intention MakeAppointment « ».

```
{
    "sessionState": {
        "intent": {},
        "activeContexts": [],
        "dialogAction": {},
        "originatingRequestId": {},
        "sessionAttributes": {},
        "runtimeHints": {
            "slotHints": {
                "MakeAppointment": {
                    "FirstName": {
                        "runtimeHintValues": [
                            {
                                "phrase": "John"
                            },
                            {
                                "phrase": "Mary"
                            }
                        ]
                    },
                    "LastName": {
                        "runtimeHintValues": [
                            {
                                "phrase": "Stiles"
                            },
                            {
                                "phrase": "Major"
                            }
                        ]
                    }
                }
            }
        }
    }
}
```

Vous pouvez également utiliser une fonction Lambda pour ajouter des indices d'exécution au cours d'une conversation. Pour ajouter des indices d'exécution, vous devez ajouter la `runtimeHints` structure à l'état de session de la réponse que votre fonction Lambda envoie à Amazon Lex V2. Pour en savoir plus, consultez Préparation du format de réponse.

Vous devez spécifier un `intentName` et valide `slotName` dans la demande, sinon Amazon Lex V2 renvoie une erreur d'exécution.

Capture de valeurs de créneaux avec des styles d'orthographe

Amazon Lex V2 fournit des emplacements intégrés pour capturer des informations spécifiques à l'utilisateur telles que le prénom, le nom de famille, l'adresse e-mail ou les identifiants alphanumériques. Par exemple, vous pouvez utiliser le `AMAZON.LastName` slot pour capturer des noms de famille tels que « Jackson » ou « Garcia ». Cependant, Amazon Lex V2 peut être confondu avec des noms de famille difficiles à prononcer ou peu courants dans une région, tels que « Xiulan ». Pour capturer de tels noms, vous pouvez demander à l'utilisateur de les saisir orthographe par lettre ou orthographe par mot.

Amazon Lex V2 propose trois styles de sélection d'emplacements que vous pouvez utiliser. Lorsque vous définissez un style d'élicitation de créneaux, cela change la façon dont Amazon Lex V2 interprète les données saisies par l'utilisateur.

Orthographe par lettre — Avec ce style, vous pouvez demander au bot d'écouter l'orthographe plutôt que la phrase entière. Par exemple, pour capturer un nom de famille tel que « Xiulan », vous pouvez demander à l'utilisateur d'épeler son nom de famille une lettre à la fois. Le bot saisira l'orthographe et transformera les lettres en un mot. Par exemple, si l'utilisateur dit « x i u l a n », le bot saisit le nom de famille sous la forme « xiulan ».

Orthographe par mot — Dans les conversations vocales, en particulier au téléphone, quelques lettres, telles que « t », « b », « p », ont un son similaire. Lorsque la capture de valeurs alphanumériques ou d'orthographe de noms entraîne une valeur incorrecte, vous pouvez demander à l'utilisateur de fournir un mot d'identification avec la lettre. Par exemple, si la réponse vocale à une demande d'identifiant de réservation est « abp123 », votre bot pourrait plutôt reconnaître l'expression « ab b 123 ». S'il s'agit d'une valeur incorrecte, vous pouvez demander à l'utilisateur de fournir l'entrée sous la forme « a comme dans alpha b comme dans boy p comme dans peter un deux trois ». Le bot résoudra l'entrée en « abp123 ».

Lorsque vous utilisez l'orthographe par mot, vous pouvez utiliser les formats suivants :

- « comme dans » (comme dans Apple)

- « pour » (a pour Apple)
- « comme » (une pomme similaire)

Par défaut : il s'agit du style naturel de capture des emplacements utilisant la prononciation des mots. Par exemple, il peut naturellement capturer des noms tels que « John Stiles ». Si aucun style d'élicitation de créneaux n'est spécifié, le bot utilise le style par défaut. Pour les types d'emplacements de `AMAZON.UKPostal` code `AMAZON.AlphaNumeric` et, le style par défaut prend en charge la saisie orthographique par lettre.

Si le nom « Xiulan » est prononcé en utilisant un mélange de lettres et de mots, tel que « x comme dans x-ray i u l comme dans lion a n », le style d'élicitation de la machine à sous doit être défini sur style. spell-by-word Le spell-by-letter style ne le reconnaîtra pas.

Vous devez créer une interface vocale qui capture les valeurs des créneaux avec un style conversationnel naturel pour une meilleure expérience. Pour les entrées qui ne sont pas correctement capturées en utilisant le style naturel, vous pouvez demander à nouveau à l'utilisateur et définir le style d'élicitation des créneaux sur ou. spell-by-letter spell-by-word

Vous pouvez utiliser spell-by-word des spell-by-letter styles pour les types de machines à sous suivants en anglais (États-Unis), anglais (Royaume-Uni) et anglais (Australie) :

- AMAZON. AlphaNumeric
- AMAZON. EmailAddress
- AMAZON. FirstName
- AMAZON. LastName
- AMAZON.UK PostalCode
- Types de machines à sous personnalisés

Activer l'orthographe

Vous activez spell-by-letter et, spell-by-word au moment de l'exécution, lorsque vous demandez des emplacements à l'utilisateur. Vous pouvez définir le style d'orthographe à l'aide de l'StartConversationopération PutSessionRecognizeTextRecognizeUtterance,, ou. Vous pouvez également activer spell-by-letter et spell-by-word utiliser une fonction Lambda.

Vous définissez le style orthographique en utilisant le `dialogAction` champ du `sessionState` champ dans la demande de l'une des opérations d'API susmentionnées ou lors de la configuration de

la réponse Lambda (voir Préparation du format de réponse pour plus d'informations). Vous ne pouvez définir le style que lorsque l'action de dialogue est de type `ElicitSlot` et que l'emplacement à obtenir est l'un des types d'emplacements pris en charge.

Le code JSON suivant indique le `dialogAction` champ défini pour utiliser le spell-by-word style :

```
"dialogAction": {
    "slotElicitationStyle": "SpellByWord",
    "slotToElicit": "BookingId",
    "type": "ElicitSlot"
}
```

Le champ `slotElicitationStyle` peut avoir la valeur `SpellByLetter`, `SpellByWord` ou `Default`. Si vous ne spécifiez aucune valeur, la valeur est définie sur`Default`.

> **Note**
>
> Vous ne pouvez pas activer spell-by-letter ou obtenir de styles par le spell-by-word biais de la console.

Exemple de code

La modification du style orthographique est généralement effectuée si la première tentative de résolution d'une valeur de slot n'a pas fonctionné. L'exemple de code suivant est une fonction Lambda Python qui utilise le spell-by-word style lors de la deuxième tentative de résolution d'un slot.

Pour utiliser l'exemple de code, vous devez avoir :

- Un bot avec une seule langue, l'anglais (GB) (en_GB).
- Une intention, « CheckAccount » avec un exemple d'énoncé, « J'aimerais vérifier mon compte ». Assurez-vous que l'option Utiliser une fonction Lambda pour l'initialisation et la validation est sélectionnée dans la section Code hooks de la définition de l'intention.
- L'intention doit comporter un emplacement, PostalCode « », du type `AMAZON.UKPostalCode` intégré.
- Alias avec la fonction Lambda définie. Pour plus d'informations, consultez Création et association d'une fonction Lambda à un alias de bot.

```
import json
import time
import os
import logging

logger = logging.getLogger()
logger.setLevel(logging.DEBUG)

# --- Helpers that build all of the responses ---

def get_slots(intent_request):
    return intent_request['sessionState']['intent']['slots']

def get_session_attributes(intent_request):
    sessionState = intent_request['sessionState']
    if 'sessionAttributes' in sessionState:
        return sessionState['sessionAttributes']
    return {}

def get_slot(intent_request, slotName):
    slots = get_slots(intent_request)
    if slots is not None and slotName in slots and slots[slotName] is not None:
        logger.debug('resolvedValue={}'.format(slots[slotName]['value']
['resolvedValues']))
        return slots[slotName]['value']['resolvedValues']
    else:
        return None

def elicit_slot(session_attributes, intent_request, slots, slot_to_elicit,
 slot_elicitation_style, message):
    return {'sessionState': {'dialogAction': {'type': 'ElicitSlot',
                                              'slotToElicit': slot_to_elicit,
                                              'slotElicitationStyle':
 slot_elicitation_style
                                              },
                             'intent': {'name': intent_request['sessionState']
['intent']['name'],
                                        'slots': slots,
                                        'state': 'InProgress'
                                        },
                             'sessionAttributes': session_attributes,
                             'originatingRequestId': 'REQUESTID'
                             },
```

```
            'sessionId': intent_request['sessionId'],
            'messages': [ message ],
            'requestAttributes': intent_request['requestAttributes']
            if 'requestAttributes' in intent_request else None
            }

def build_validation_result(isvalid, violated_slot, slot_elicitation_style,
 message_content):
    return {'isValid': isvalid,
            'violatedSlot': violated_slot,
            'slotElicitationStyle': slot_elicitation_style,
            'message': {'contentType': 'PlainText',
            'content': message_content}
            }

def GetItemInDatabase(postal_code):
    """
    Perform database check for transcribed postal code. This is a no-op
    check that shows that postal_code can't be found in the database.
    """
    return None

def validate_postal_code(intent_request):

    postal_code = get_slot(intent_request, 'PostalCode')

    if GetItemInDatabase(postal_code) is None:
        return build_validation_result(
            False,
            'PostalCode',
            'SpellByWord',
            "Sorry, I can't find your information. " +
            "To try again, spell out your postal " +
            "code using words, like a as in apple."
        )
    return {'isValid': True}

def check_account(intent_request):
    """
    Performs dialog management and fulfillment for checking an account
    with a postal code. Besides fulfillment, the implementation for this
    intent demonstrates the following:
    1) Use of elicitSlot in slot validation and re-prompting.
    2) Use of sessionAttributes to pass information that can be used to
```

```
        guide a conversation.
    """
    slots = get_slots(intent_request)
    postal_code = get_slot(intent_request, 'PostalCode')
    session_attributes = get_session_attributes(intent_request)

    if intent_request['invocationSource'] == 'DialogCodeHook':
        # Validate the PostalCode slot. If any aren't valid,
        # re-elicit for the value.
        validation_result = validate_postal_code(intent_request)
        if not validation_result['isValid']:
            slots[validation_result['violatedSlot']] = None
            return elicit_slot(
                session_attributes,
                intent_request,
                slots,
                validation_result['violatedSlot'],
                validation_result['slotElicitationStyle'],
                validation_result['message']
            )

        return close(
            intent_request,
            session_attributes,
            'Fulfilled',
            {'contentType': 'PlainText',
             'content': 'Thanks'
             }
        )

def close(intent_request, session_attributes, fulfillment_state, message):
    intent_request['sessionState']['intent']['state'] = fulfillment_state
    return {
        'sessionState': {
            'sessionAttributes': session_attributes,
            'dialogAction': {
                'type': 'Close'
            },
            'intent': intent_request['sessionState']['intent'],
            'originatingRequestId': 'xxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
        },
        'messages': [ message ],
        'sessionId': intent_request['sessionId'],
```

```
        'requestAttributes': intent_request['requestAttributes'] if 'requestAttributes'
 in intent_request else None
    }

# --- Intents ---

def dispatch(intent_request):
    """
    Called when the user specifies an intent for this bot.
    """
    intent_name = intent_request['sessionState']['intent']['name']
    response = None

    # Dispatch to your bot's intent handlers
    if intent_name == 'CheckAccount':
        response = check_account(intent_request)

    return response

# --- Main handler ---

def lambda_handler(event, context):
    """
    Route the incoming request based on the intent.

    The JSON body of the request is provided in the event slot.
    """

    # By default, treat the user request as coming from
    # Eastern Standard Time.
    os.environ['TZ'] = 'America/New_York'
    time.tzset()

    logger.debug('event={}'.format(json.dumps(event)))
    response = dispatch(event)
    logger.debug("response={}".format(json.dumps(response)))

    return response
```

Surveillance des performances des robots

La surveillance est importante pour garantir la fiabilité, la disponibilité et les performances de vos chatbots Amazon Lex V2. Cette rubrique décrit l'utilisation des journaux de conversation pour surveiller les conversations entre vos utilisateurs et vos chatbots, l'utilisation des statistiques relatives aux énoncés pour déterminer les énoncés que vos robots détectent et oublient, ainsi que la façon d'utiliser Amazon CloudWatch Logs et de surveiller AWS CloudTrail Amazon Lex V2. Il décrit également le temps d'exécution d'Amazon Lex V2 et les métriques d'association de canaux.

Utilisez ces outils et indicateurs pour comprendre les orientations et les mesures que vous pouvez prendre pour améliorer les performances de vos robots.

Rubriques

Mesurer les performances de l'entreprise avec Analytics

Avec Analytics, vous pouvez évaluer les performances de votre bot à l'aide de mesures liées au taux de réussite et d'échec des interactions de vos robots avec les clients. Vous pouvez également visualiser les modèles de flux de conversation entre votre bot et les clients. Analytics rationalise votre expérience en résumant ces indicateurs sous forme de graphiques et de tableaux. Analytics fournit des outils qui vous aident à filtrer les résultats afin d'identifier les problèmes liés aux intentions, aux créneaux, aux énoncés et aux conversations. Vous pouvez utiliser ces données pour itérer et améliorer votre bot afin de créer une meilleure expérience client.

> **Note**
>
> Pour qu'un utilisateur puisse accéder à Analytics, la politique Politique gérée par AWS : AmazonLexFullAccess ou une politique personnalisée incluant des autorisations d'API d'analyse doit être attachée à son rôle IAM. Consultez Gestion des autorisations d'accès pour les analyses pour plus de détails sur la façon de gérer les autorisations des utilisateurs à l'aide d'une politique personnalisée. Si le Politique gérée par AWS : AmazonLexReadOnly est attaché au rôle IAM d'un client, une erreur affiche les autorisations manquantes que vous

devez ajouter au rôle IAM de l'utilisateur pour qu'il puisse accéder aux tableaux de bord Analytics.

Pour accéder à Analytics

1. Connectez-vous à la console Amazon Lex V2 AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/lexv2/home.
2. Dans le volet de navigation, sous Bots, sélectionnez le bot que vous souhaitez afficher dans les analyses.
3. Sélectionnez la section sous Analytics que vous souhaitez consulter.

Rubriques

Définitions clés

Cette rubrique fournit des définitions clés qui vous aideront à interpréter les analyses de votre bot. Ces définitions sont liées aux performances de votre bot dans quatre contextes : les intentions, les machines à sous, les conversations et les énoncés. Les champs suivants sont pertinents pour de nombreux indicateurs de performance :

- `state`Champ de l'`Intent`objet.
- `type`Champ de l'`dialogAction`objet au sein de l'SessionStateobjet.

Intentions

Amazon Lex V2 classe les intentions de la manière suivante :

- Succès — Le bot a rempli son objectif avec succès. L'une des situations suivantes est vraie :
  - L'intention `state` est `ReadyForFulfillment` et `dialogAction` est`Close`. `type`
  - L'intention `state` est `Fulfilled` et `dialogAction` est`Close`. `type`
- Échec : le bot n'a pas répondu à son intention. État d'intention. L'une des situations suivantes est vraie :
  - L'intention `state` est `Failed` et est `type` toujours `dialogAction` `Close` (par exemple, l'utilisateur a refusé l'invite de confirmation).
  - Le bot passe en mode `AMAZON.FallbackIntent` avant que l'intention ne soit terminée.
- Commuté : le bot reconnaît une intention différente et passe à cette intention à la place, avant que l'intention initiale ne soit considérée comme un succès ou un échec.
- Abandonné — Le client ne répond pas avant que l'intention ne soit qualifiée de réussite ou d'échec.

Emplacements

Amazon Lex V2 classe les emplacements de la manière suivante :

- Succès : le bot a rempli le créneau et est passé avec succès à un autre emplacement ou à l'étape de confirmation.
- Échec : le bot n'a pas pu remplir le créneau, même après avoir atteint le nombre maximum de tentatives.
- Abandonné : le client ne répond pas ou passe à une autre intention avant que le créneau ne soit considéré comme un succès ou un échec.

Conversations

Lorsqu'un client passe un appel d'exécution à Amazon Lex V2, il fournit un `sessionId`et Amazon Lex V2 génère un `originatingRequestId`. Si le client ne répond pas dans le délai d'expiration de session (`idleSessionTTLInSeconds`) que vous avez défini pour le bot, la session expire. Si un client revient à la session en l'utilisant`sessionId`, Amazon Lex V2 en génère une nouvelle`originatingRequestId`.

Dans le domaine de l'analytique, une conversation est une combinaison unique de a `sessionId` et de`originatingRequestId`. Amazon Lex V2 classe les conversations de la manière suivante :

- Succès — L'intention finale de la conversation est considérée comme un succès.

- Échec : l'intention finale de la conversation est un échec. La conversation échoue également si Amazon Lex V2 utilise par défaut leAMAZON.FallbackIntent.
- Abandonné : le client ne répond pas avant que la conversation ne soit considérée comme un succès ou un échec.

Énoncés

Amazon Lex V2 classe les énoncés de la manière suivante :

- Détecté : Amazon Lex V2 reconnaît l'énoncé comme une tentative d'invoquer une intention configurée pour un bot.
- Manqué : Amazon Lex V2 ne reconnaît pas l'énoncé.

Filtrage des résultats

En haut de chaque page, vous pouvez filtrer les résultats pour les analyses de votre bot.
Options de filtrage pour les analyses.

Vous pouvez filtrer selon les paramètres suivants :

- Heure : vous pouvez filtrer les résultats par plage de temps relative ou absolue. Lorsque vous sélectionnez une heure de début et une heure de fin, Amazon Lex V2 récupère les conversations qui ont débuté après l'heure de début et qui se sont terminées avant l'heure de fin.
  - Plage relative : sélectionnez 1d pour voir les résultats du jour précédent, 1w pour la semaine dernière ou 1 m pour le mois dernier.

    Pour plus d'options, sélectionnez Personnalisé et choisissez une durée dans le menu Plage relative. Pour mieux contrôler la durée, sélectionnez Plage personnalisée, entrez un nombre dans le champ Durée et choisissez une unité de temps dans le menu déroulant.
  - Plage absolue : sélectionnez Personnalisé et choisissez le menu Plage absolue pour filtrer les conversations dans une plage de temps que vous spécifiez. Vous pouvez choisir une date de début et de fin sur le calendrier ou la saisir au format AAAA/MM/JJ.

> **Note**
>
> La durée maximale pendant laquelle vous pouvez voir les résultats est de 30 jours.

- Filtres de robots : pour filtrer par localisation, alias et version de votre bot, sélectionnez les menus déroulants intitulés Toutes les langues, Tous les alias et Toutes les versions.
- Modalité — Sélectionnez l'icône représentant un engrenage et choisissez le menu déroulant Modalité pour choisir d'afficher les résultats pour le langage vocal ou le texte.
- Chaîne : sélectionnez l'icône en forme de roue dentée et choisissez le menu déroulant Chaîne pour choisir la chaîne pour laquelle vous souhaitez afficher les résultats. Pour plus d'informations sur l'intégration des canaux, consultez Intégration d'un bot Amazon Lex V2 à une plateforme de messagerie et les centres de contact Amazon Connect

Vue d'ensemble : résumé des performances de votre bot

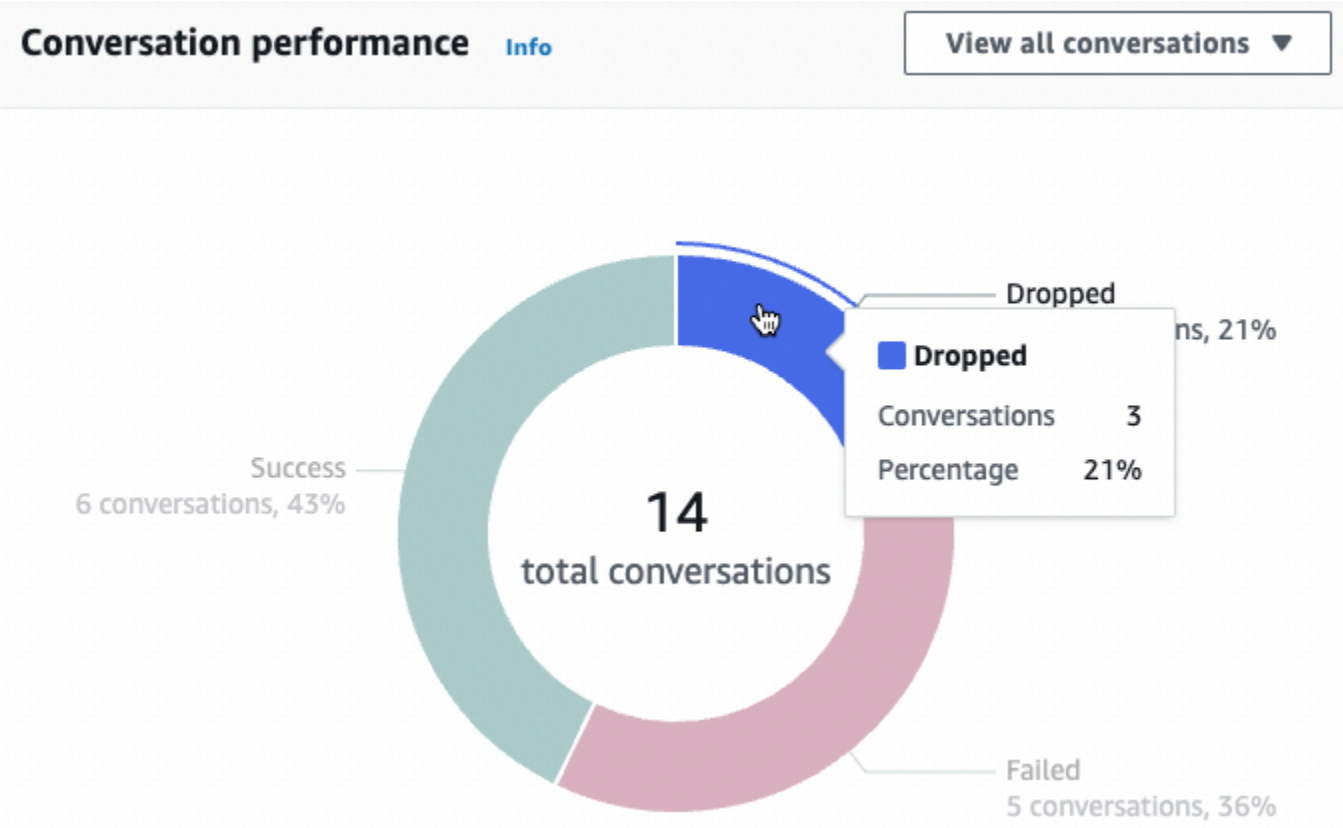
La page de présentation résume les performances de votre bot en matière de conversations, de reconnaissance des énoncés et d'utilisation intentionnelle. L'aperçu comprend les sections suivantes :

- Performance de conversation
- Taux de reconnaissance des énoncés
- Historique des performances des conversations
- Les 5 meilleures intentions utilisées
- Les 5 meilleures intentions infructueuses

Performance de conversation

Utilisez ce graphique pour suivre le nombre et le pourcentage de conversations classées comme réussies, échouées ou abandonnées. Pour accéder à une liste de conversations, sélectionnez Afficher toutes les conversations pour afficher un menu déroulant. Vous pouvez choisir d'afficher la liste de toutes les conversations des utilisateurs avec le bot ou de filtrer les conversations ayant un résultat spécifique (réussite, échec ou échec). Ces liens vous mènent à la sous-section Conversations du tableau de bord des conversations. Pour plus d’informations, consultez Conversations.

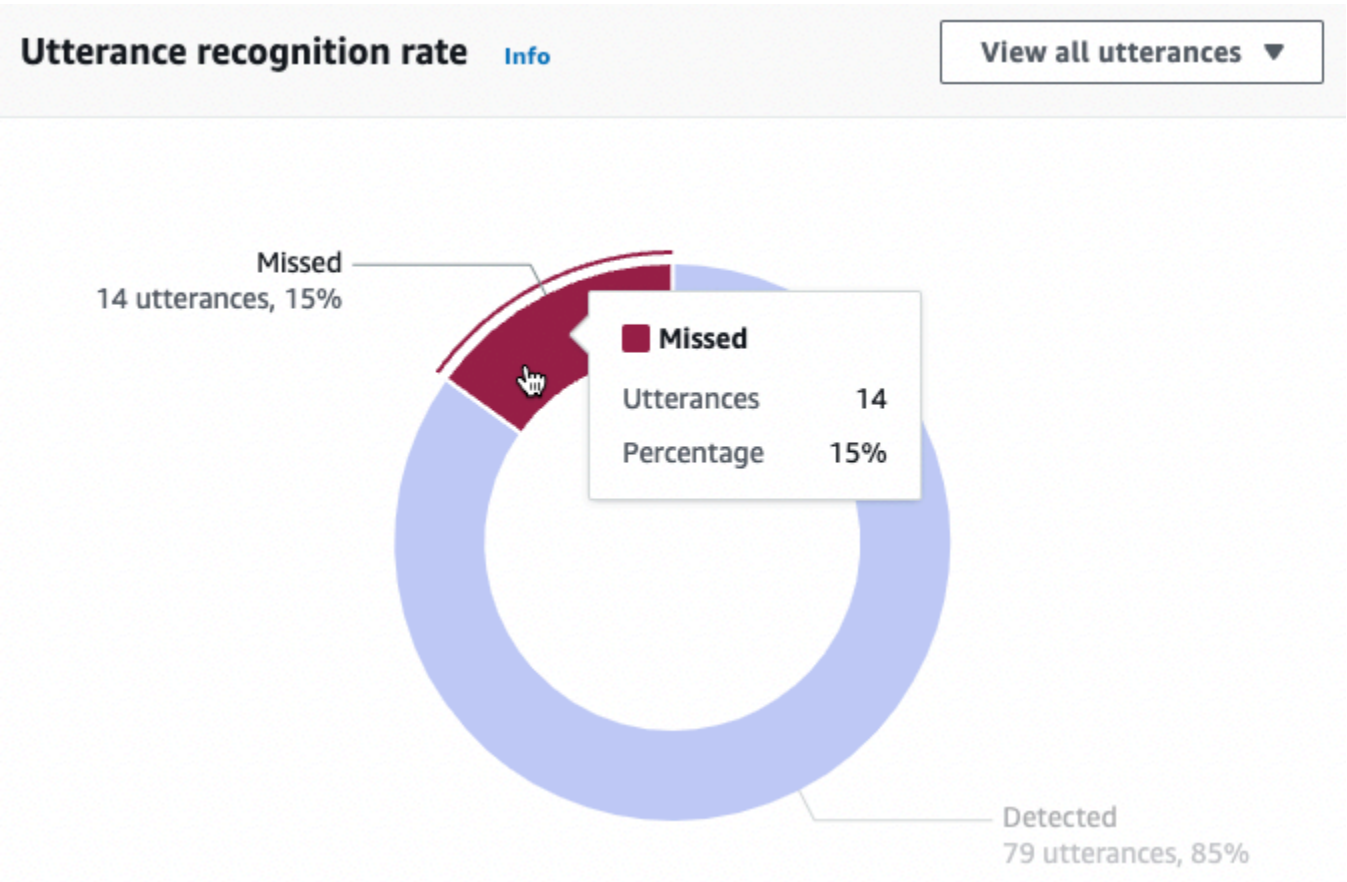
Pour afficher une case indiquant le nombre et le pourcentage de conversations ayant donné ce résultat, passez le curseur sur un segment du graphique, comme dans l'image suivante.



Taux de reconnaissance des énoncés

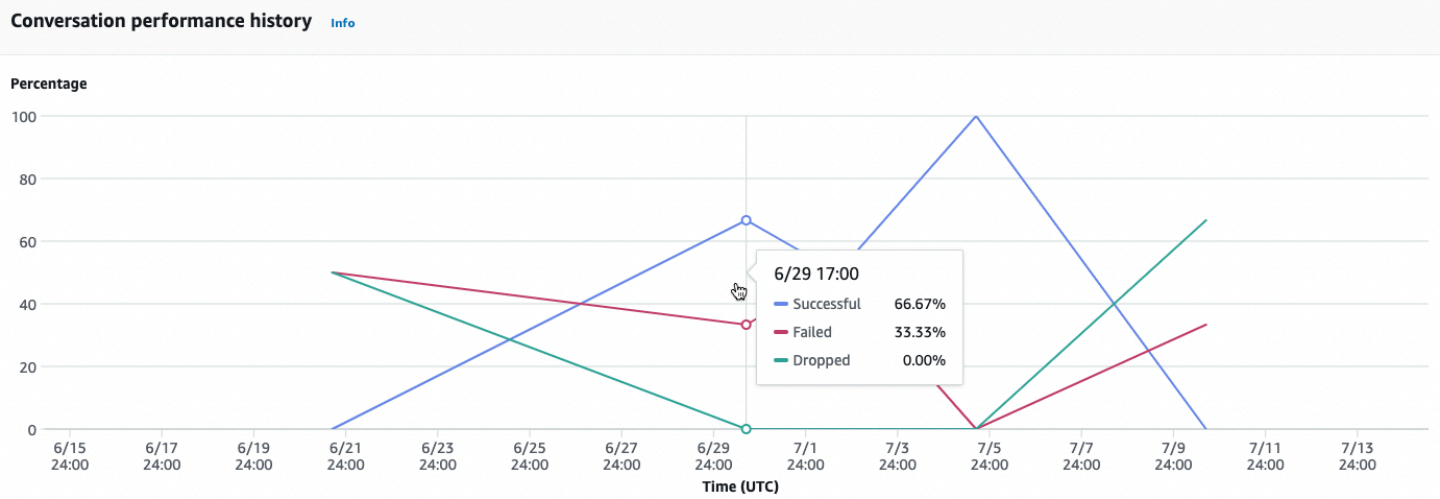
Utilisez ce tableau pour suivre le nombre et le pourcentage d'énoncés détectés et oubliés par votre bot. Pour accéder à une liste d'énoncés, sélectionnez Afficher les énoncés pour afficher un menu déroulant. Vous pouvez choisir de consulter la liste de tous les énoncés de l'utilisateur ou de filtrer les énoncés ayant un résultat spécifique (manqués ou détectés). Ces liens vous mènent à la sous-section Reconnaissance des énoncés du tableau de bord des performances. Pour plus d'informations, voir Afficher les énoncés pour accéder à. Reconnaissance des énoncés

Pour afficher une case indiquant le nombre et le pourcentage d'énoncés, passez le pointeur de la souris sur un segment du graphique, comme illustré dans l'image suivante.



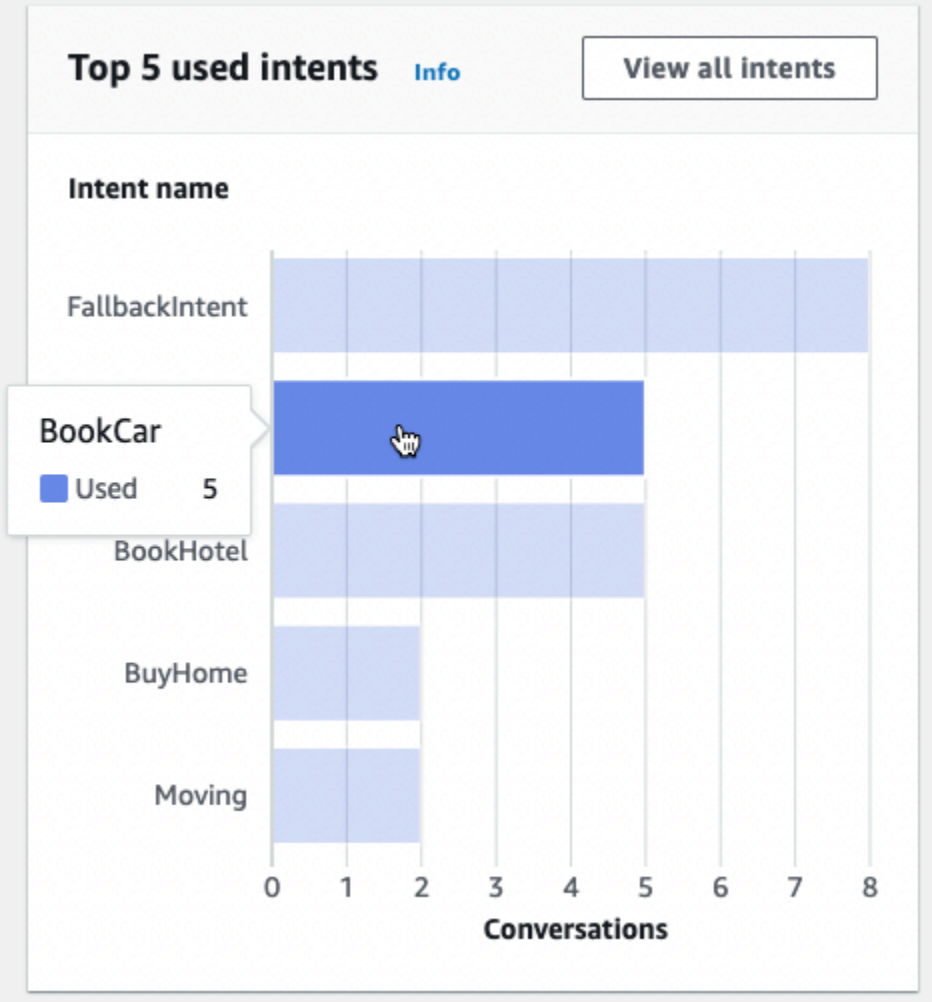
Historique des performances des conversations

Utilisez ce graphique pour suivre le pourcentage de conversations classées comme réussies, échouées ou abandonnées sur la période que vous avez définie dans les filtres. Pour voir le pourcentage de conversations ayant un résultat spécifique dans un intervalle de temps, passez le curseur sur cet intervalle, comme dans l'image suivante.



Les 5 meilleures intentions utilisées

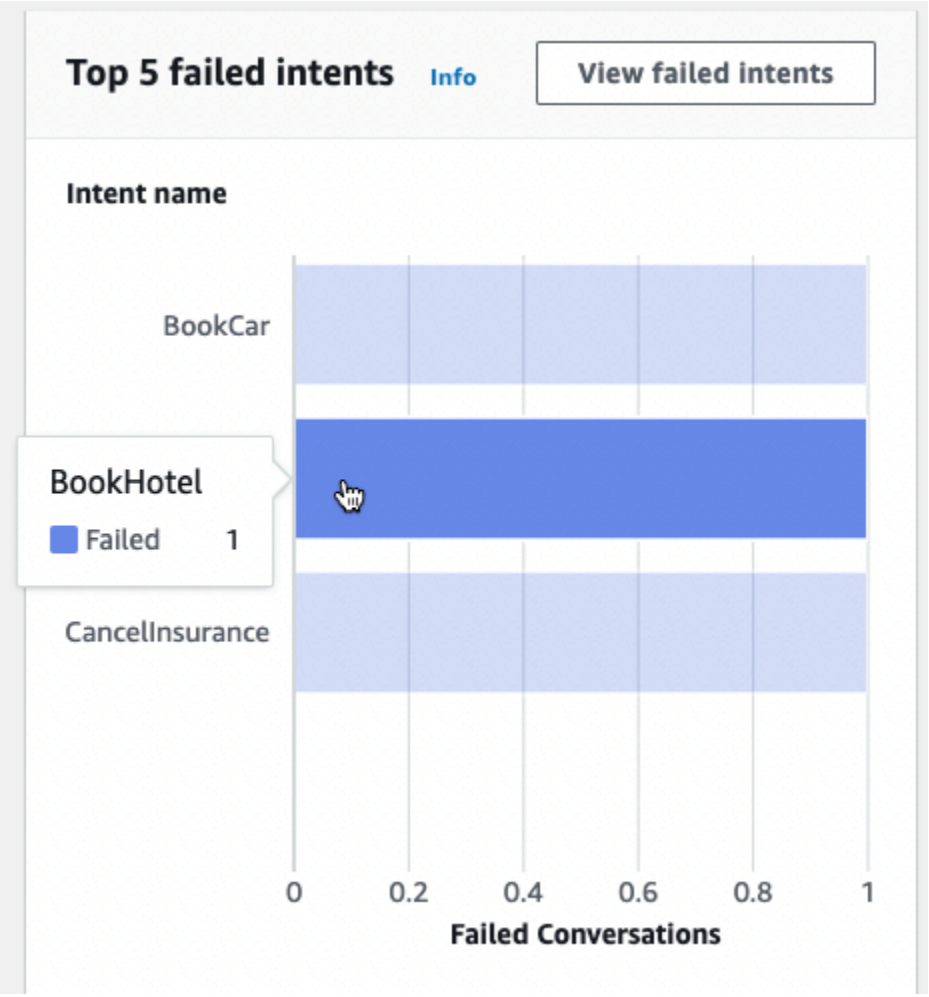
Utilisez ce tableau pour identifier les cinq principales intentions utilisées par les clients avec votre bot. Passez la souris sur une barre pour voir le nombre de fois que votre bot a reconnu cette intention, comme dans l'image suivante.



Sélectionnez Afficher toutes les intentions pour accéder à la sous-section Performances des intentions du tableau de bord des performances, où vous pouvez consulter les mesures relatives aux performances de votre bot en termes de réalisation des intentions. Pour plus d'informations, consultez Intention et performance.

Les 5 meilleures intentions infructueuses

Utilisez ce tableau pour identifier les cinq principales intentions que votre bot n'a pas respectées (voir Intentions la définition d'une intention infructueuse). Passez la souris sur une barre pour voir le nombre de fois où votre bot n'a pas répondu à cette intention, comme dans l'image suivante.



Sélectionnez Afficher les intentions infructueuses pour accéder à la sous-section Performances des intentions du tableau de bord des performances, où vous pouvez consulter les mesures relatives aux intentions que votre bot n'a pas respectées. Pour plus d'informations, consultez Intention et performance.

Tableau de bord des conversations : résumé des conversations de vos robots

Le tableau de bord des conversations permet de visualiser les statistiques relatives aux conversations des clients (voir Conversations la définition d'une conversation) avec votre bot.

Le résumé contient les informations suivantes sur les conversations des utilisateurs avec votre bot. Les chiffres sont calculés en fonction des paramètres du filtre.

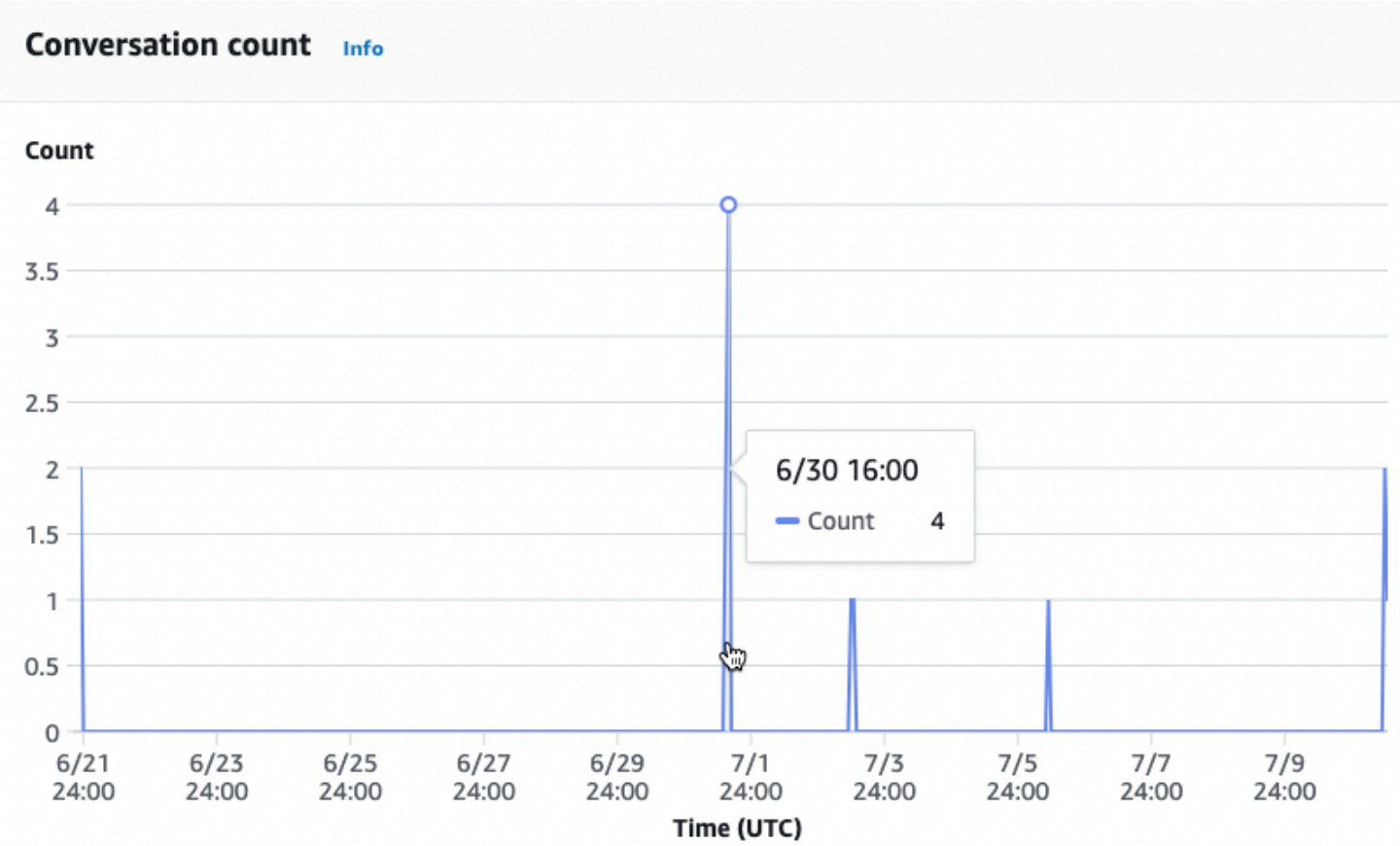
- Nombre total de conversations : nombre total de conversations avec le bot.

- Durée moyenne des conversations : durée moyenne des conversations des utilisateurs avec le bot, en minutes et en secondes. Le format est mm:ss.
- Nombre moyen de tours par conversation : nombre moyen de tours que dure une conversation.

Les sections Nombre de conversations et Nombre de messages contiennent chacune un graphique qui indique le nombre de conversations et de messages, respectivement, sur la période que vous spécifiez dans vos filtres. Passez le curseur sur un segment de temps pour voir le nombre de conversations ou de messages contenus dans ce segment. La taille du segment temporel dépend de la plage de temps que vous spécifiez :

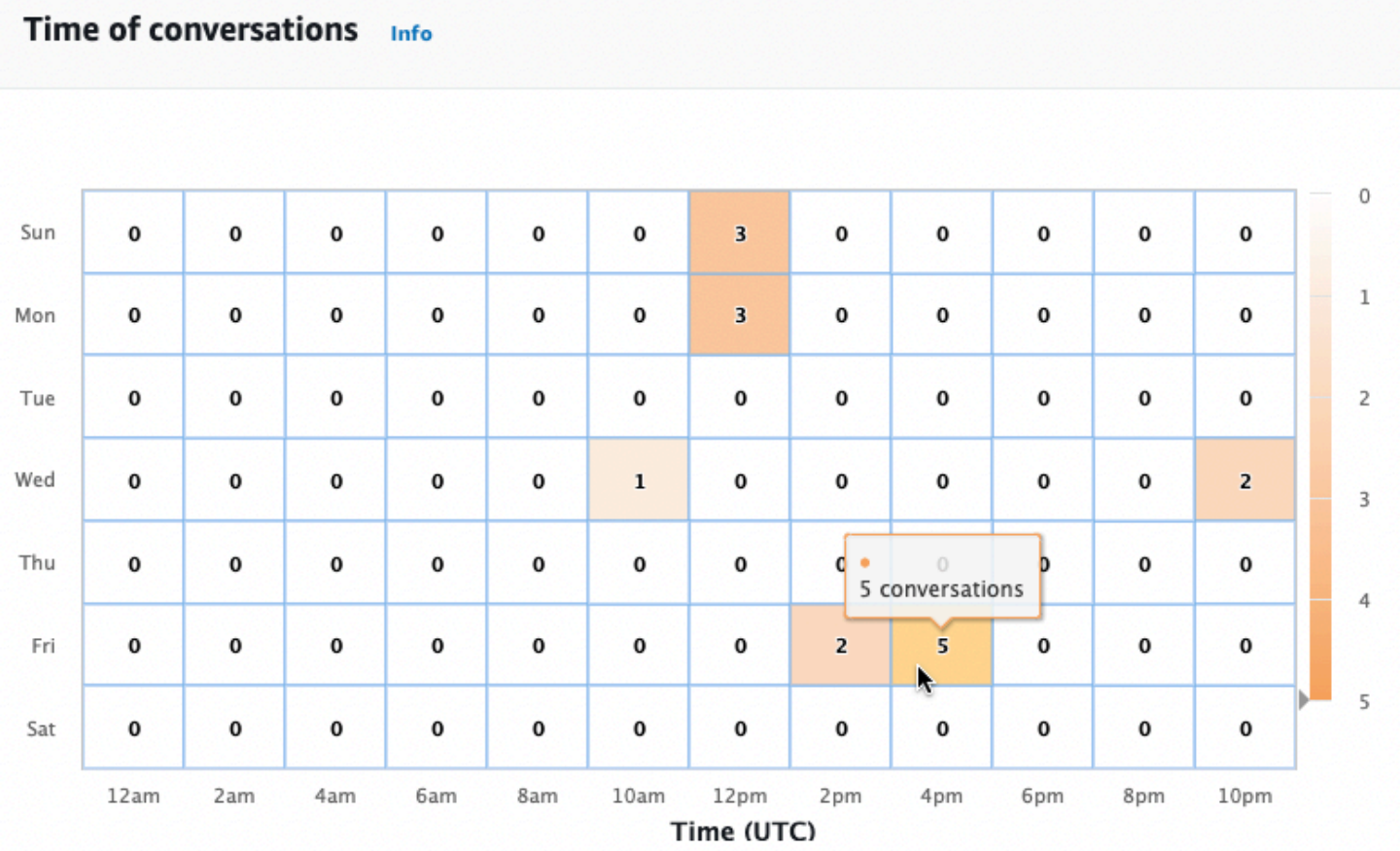
- Moins d'une semaine : le décompte est affiché pour chaque heure.
- 1 semaine ou plus — Le nombre est affiché pour chaque jour.

Vous trouverez un exemple du comportement de survol dans l'image suivante.



La section Durée des conversations indique le nombre de conversations qui ont eu lieu entre votre bot et les clients au cours de chaque intervalle de deux heures, chaque jour de la semaine, dans la

plage de temps que vous spécifiez dans les filtres. Les cellules plus foncées indiquent les heures auxquelles le plus grand nombre de conversations ont eu lieu. Passez le pointeur de la souris sur une cellule pour afficher le nombre de conversations au cours des 2 heures à compter de ce créneau horaire. Par exemple, l'action présentée dans l'image suivante montre le nombre de conversations ayant lieu entre 16 h 00 et 18 h 00 UTC.



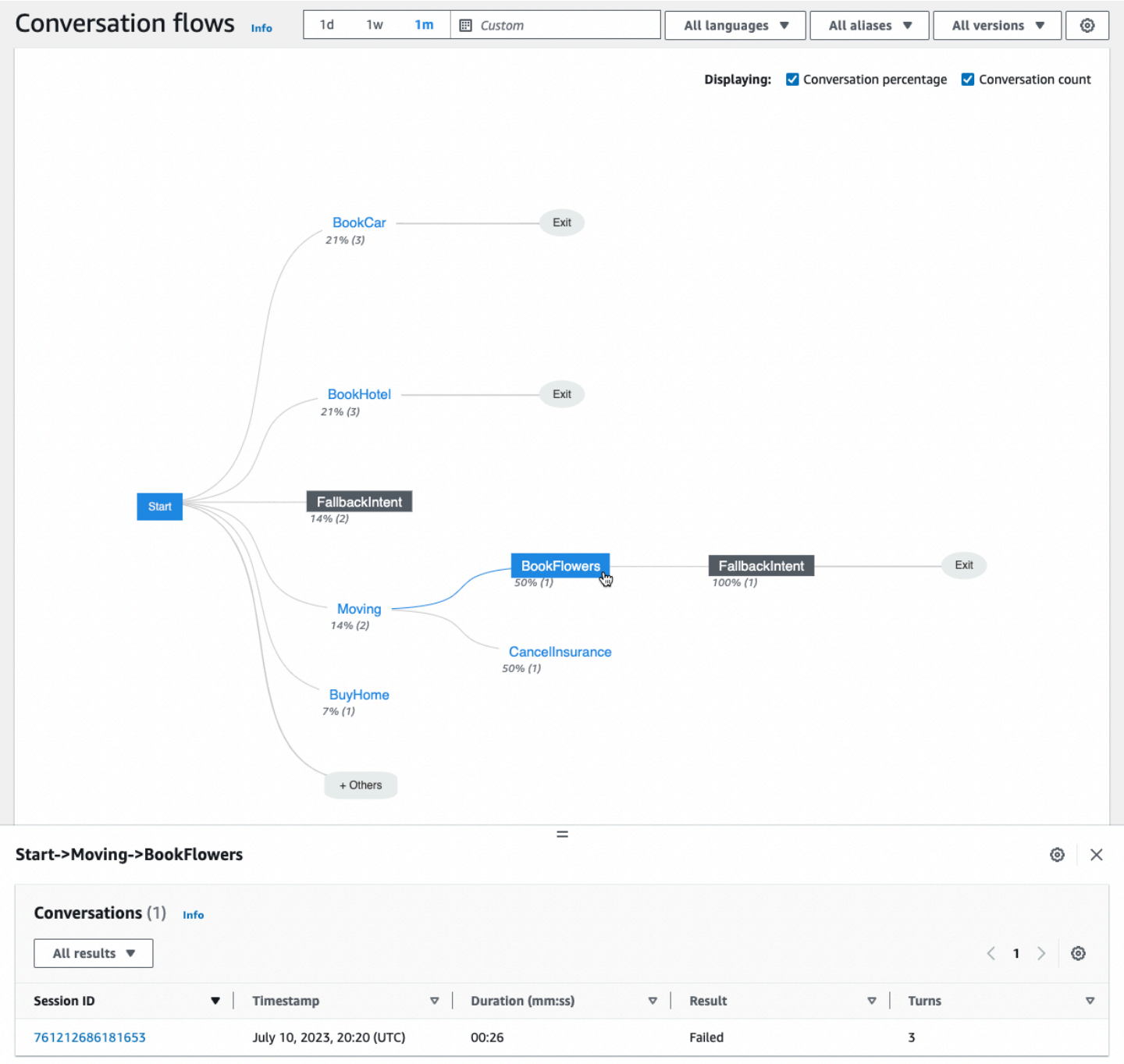
Le tableau de bord de conversation contient deux outils, les flux de conversation et les conversations. Accédez à un outil en le sélectionnant sous Tableau de bord de conversation dans le volet de navigation de gauche.

Flux de conversation

Utilisez les flux de conversation pour visualiser l'ordre des intentions des clients lors des conversations avec votre bot. Sous chaque intention se trouvent le pourcentage et le nombre de conversations qui ont invoqué cette intention à ce stade de la conversation. Vous pouvez modifier le pourcentage et le décompte en sélectionnant Pourcentage de conversation et Nombre de conversations en haut. Par défaut, les cinq intentions les plus courantes à ce stade de la conversation sont affichées par ordre décroissant de fréquence. Sélectionnez + Autres pour afficher toutes les intentions.

Choisissez une intention à étendre à une nouvelle colonne de branches qui affiche une liste des intentions prises à ce stade de la conversation, triées par fréquence décroissante.

Lorsque vous sélectionnez un nœud dans le flux de conversation, vous pouvez agrandir la fenêtre ci-dessous pour afficher la liste des conversations qui ont suivi cet ordre d'intention. Choisissez l'ID de session correspondant à une conversation pour afficher les détails de cette conversation. L'image suivante montre un flux de conversation et une fenêtre de conversation étendue en bas.



Conversations

L'outil Conversations affiche une liste des conversations pour votre bot. Vous pouvez sélectionner une colonne pour la trier par ordre croissant ou décroissant.

Pour filtrer les conversations par résultat, sélectionnez Tous les résultats, puis choisissez Succès, Échec ou Abandonné.

Pour filtrer les conversations par durée

1. Sélectionnez la barre de recherche marquée Filtrer les conversations par durée
2. Définissez le filtre de l'une des manières suivantes :
   - Utilisez les options prédéfinies.
     a. Sélectionnez Durée.
     b. Choisissez entre les opérateurs = (égal), > (supérieur à) et < (inférieur à).
     c. Choisissez une durée.
   - Entrez une entrée au format « Durée {opérateur} {nombre} sec ». Par exemple, pour rechercher toutes les conversations d'une durée supérieure à 30 secondes, entrez**`Duration > 30 sec`**. Spécifiez la durée en secondes.

Pour afficher des informations détaillées sur la session, notamment les métadonnées, l'utilisation intentionnelle et une transcription, sélectionnez l'ID de session d'une conversation.

> **Note**
>
> Comme une conversation est une combinaison unique d'un `sessionId` et`originatingRequestId`, la même chose `sessionId` peut apparaître plusieurs fois dans le tableau.

La section Détails contient les métadonnées suivantes :

- Horodatage — Spécifie la date et l'heure de début de la conversation. L'heure est au format hh:mm:ss.
- Durée — Spécifie la durée de la conversation au format mm:ss. La durée n'inclut pas le délai d'expiration de la session (`idleSessionTTLInSeconds`).

- Résultat — Spécifie si la conversation a été classée comme réussie, échouée ou abandonnée. Voir Conversations pour plus de détails sur ces résultats.
- Mode — Spécifie si la conversation était `SpeechText`, ou `DTMF` (touches tactiles appuyées sur le clavier). Une conversation composée de plusieurs modes est`Multimode`.
- Canal — Spécifie le canal sur lequel la conversation a eu lieu, le cas échéant. veuillez consulter Intégration d'un bot Amazon Lex V2 à une plateforme de messagerie.
- Langue — Spécifie la langue du bot.

Les intentions que le bot a suscitées au cours de la conversation sont indiquées ci-dessous. Sélectionnez Accéder à l'intention pour accéder à cette intention dans l'éditeur d'intention. Sélectionnez Snap to transcript pour faire défiler automatiquement la transcription jusqu'à la première instance dans laquelle le bot a suscité l'intention.

Sélectionnez la flèche droite à côté du nom de l'intention pour afficher les détails sur les emplacements sélectionnés pour l'intention, y compris les noms des emplacements, la valeur obtenue par le bot pour chaque emplacement et le nombre de fois que le bot a tenté d'obtenir chaque emplacement.

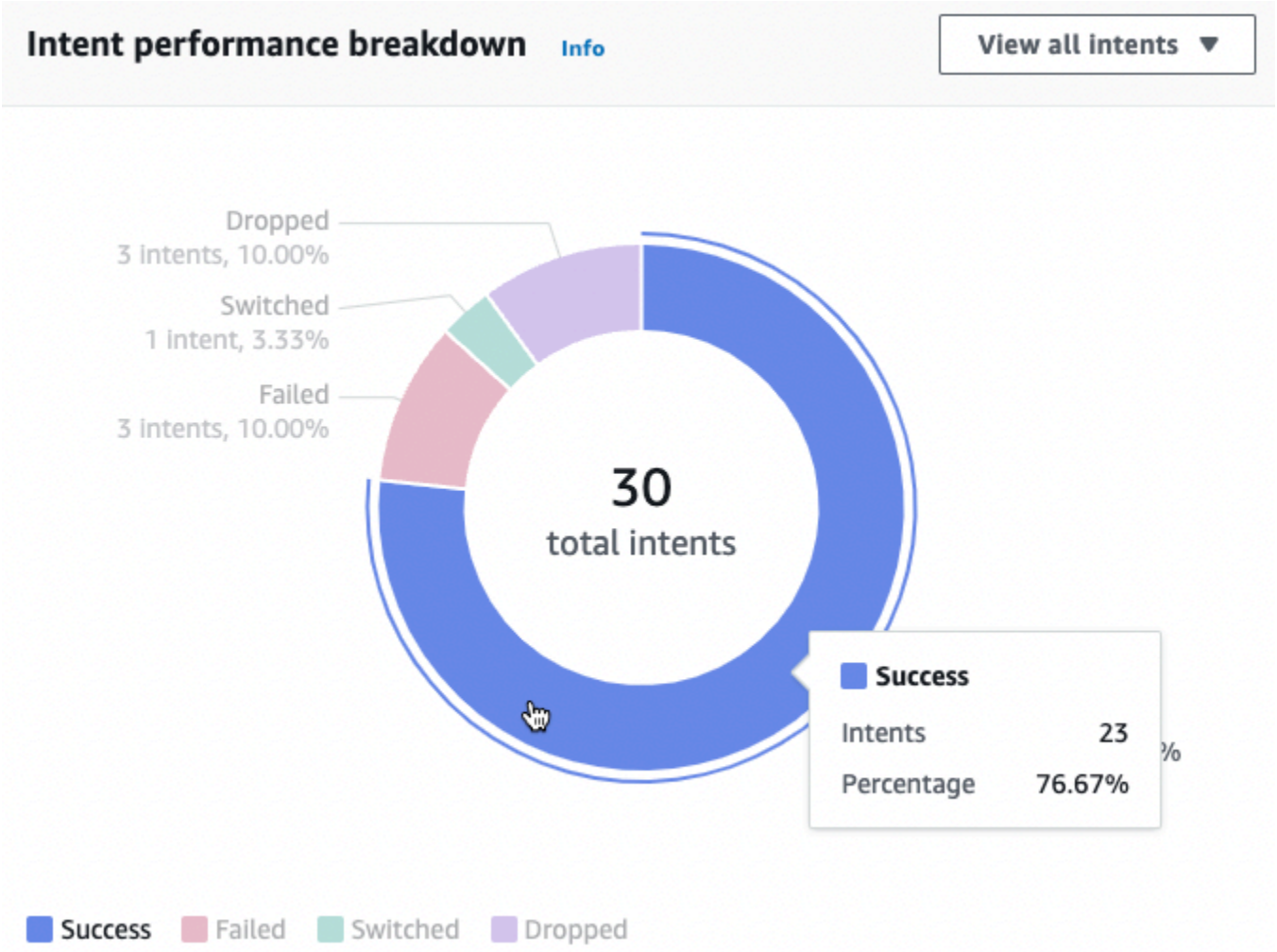
La transcription vous permet de passer en revue les énoncés de la conversation et le comportement de votre bot lorsqu'il s'agit d'obtenir des intentions et des créneaux. Les énoncés de l'utilisateur sont affichés sur la gauche et les énoncés du bot sont affichés sur la droite. Utilisez la barre de recherche intitulée Filtrer les transcriptions dans cette session pour trouver du texte dans la transcription. À côté de l'affichage : trois informations apparaissent sous chaque tour de conversation que vous pouvez choisir d'afficher ou non :

- Horodatage — Spécifie l'heure de l'énoncé.
- État d'intention — Spécifie l'intention que le bot suscite lors d'un énoncé et le résultat de cette intention, le cas échéant. Les états d'intention suivants sont possibles :
  - Intention invoquée : *`nom de l'intention`* — Le bot a identifié une intention invoquée par le client.
  - Intention modifiée : *`nom de l'intention`* — Le bot est passé à une intention différente en fonction de l'énoncé.
  - *`nom de l'intention`* : Success — Le bot a atteint son objectif.
- État de l'emplacement — Spécifie le créneau que le bot recherche lors d'un énoncé, le cas échéant, et la valeur fournie par le client.

Tableau de bord des performances : résumé des indicateurs d'intention et d'énoncé de votre bot

Dans le tableau de bord des performances, vous pouvez consulter des informations détaillées sur les performances de votre bot en matière de réalisation des intentions et de reconnaissance des énoncés.

La section Répartition des performances des intentions affiche le nombre total de fois que votre bot a invoqué une intention et indique le nombre et le pourcentage de fois où les intentions ont été classées comme un succès, un échec, une tentative abandonnée ou une modification. Voir Intentions pour une explication de ces définitions. Passez le curseur sur un segment du graphique pour afficher une case indiquant le nombre et le pourcentage de conversations ayant abouti à ce résultat, comme dans l'image suivante.



Sélectionnez Afficher toutes les intentions pour afficher un menu déroulant dans lequel vous pouvez choisir d'afficher la liste des intentions obtenues par le bot. Vous pouvez également choisir d'afficher

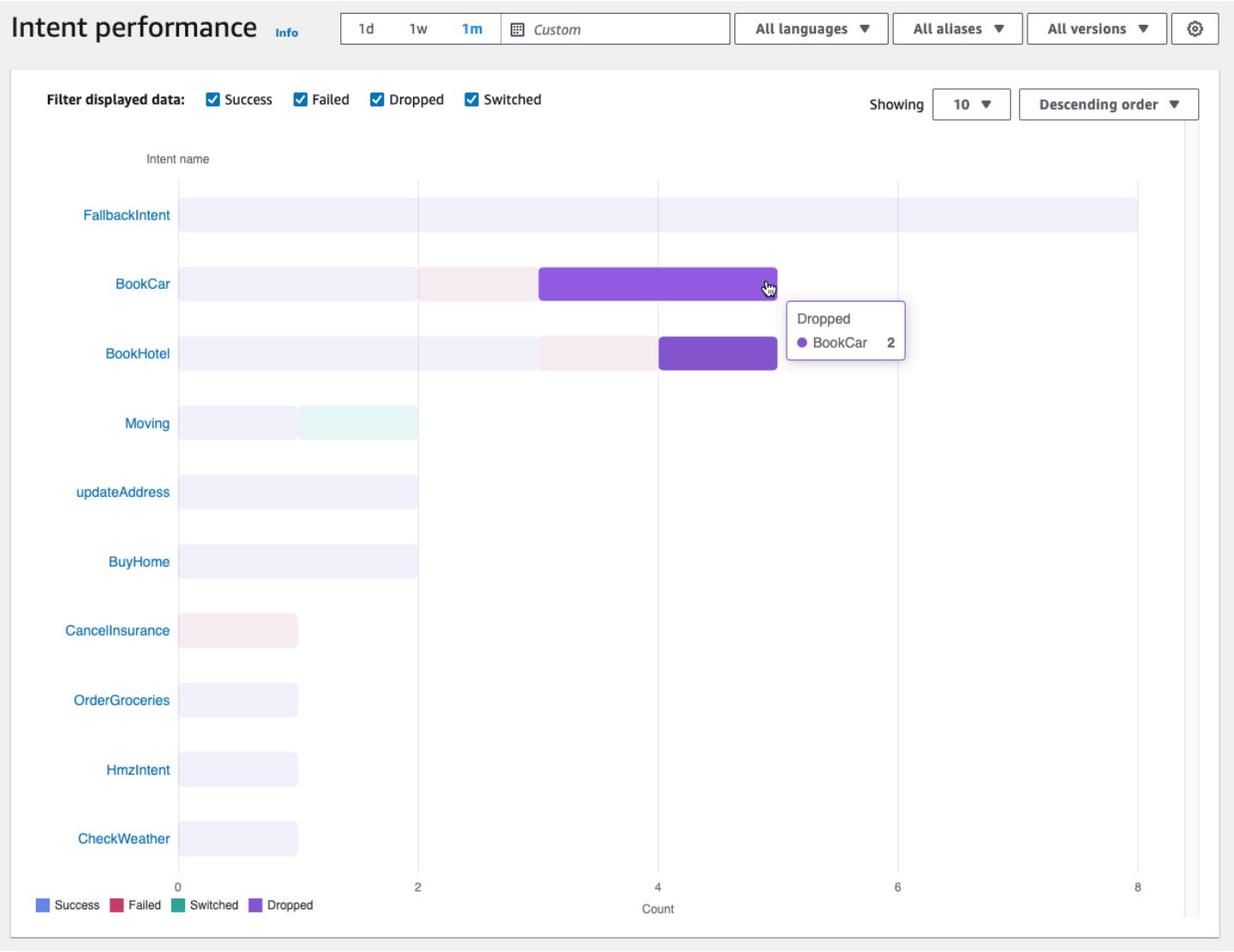
les intentions avec un résultat spécifique (réussite, échec, abandon ou changement). Ces liens vous redirigent vers la sous-section Performance des intentions du tableau de bord des performances. Pour plus d'informations, consultez Intention et performance.

La section Reconnaissance des énoncés récapitule le nombre d'énoncés qui ont été manqués et détectés. Sélectionnez Afficher les détails pour accéder à la liste des énoncés du bot. Choisissez le numéro sous Énoncés manqués pour voir la liste des énoncés manqués et le numéro sous Énoncés détectés pour voir la liste des énoncés détectés pour le bot. Pour plus d'informations, consultez Reconnaissance des énoncés.

Sélectionnez Performances des intentions et Reconnaissance des énoncés dans le tableau de bord des performances dans la barre latérale gauche pour afficher les détails sur les intentions et les énoncés de votre bot.

Intention et performance

Ce tableau de bord résume les performances des intentions utilisées avec votre bot par ordre décroissant de fréquence. La barre située à côté de chaque intention permet de visualiser le nombre de fois où l'intention a été classée comme réussie, échouée, abandonnée et modifiée. Voir Intentions pour une explication de ces définitions. Passez le pointeur de la souris sur un segment de la barre pour voir le nombre de conversations utilisant cette intention et le résultat obtenu, comme dans l'image suivante :



> **Note**
>
> Le tableau de bord affiche les 1 000 meilleurs résultats pour un ensemble de paramètres de filtre. Pour obtenir des résultats plus ciblés, configurez les paramètres du filtre granulaire.

En haut du graphique, vous pouvez changer les statuts d'intention que vous souhaitez afficher à l'aide des cases à cocher Réussite, Échec, Abandonné et Commuté.

Sélectionnez les menus déroulants situés à droite de Afficher pour ajuster le nombre d'intentions à afficher et déterminer s'il faut afficher les intentions par ordre croissant ou décroissant de fréquence.

Sélectionnez un nom d'intention pour accéder à une page qui affiche trois graphiques : répartition des performances par intention, performance des emplacements et commutateurs d'intention.

La section Répartition des performances par intention affiche le nombre total de fois que le bot a utilisé l'intention et ventile le nombre et le pourcentage de fois où l'exécution de l'intention a été classée comme un succès, un échec, un échec, un échec et un changement. Voir Intentions pour une explication de ces définitions. Passez la souris sur un segment du graphique pour voir le nombre et le pourcentage de fois où la réalisation de l'intention a produit ce résultat.

La section Performances des emplacements affiche les mesures relatives aux emplacements correspondant à l'intention actuelle. Pour trier par colonne, sélectionnez cette colonne une fois pour trier par ordre croissant et deux fois pour la trier par ordre décroissant. Vous pouvez utiliser la barre de recherche pour trouver un emplacement spécifique ou utiliser les boutons de numéro de page pour parcourir les emplacements.

> **Note**
>
> Le tableau de bord affiche les 1 000 meilleurs résultats pour un ensemble de paramètres de filtre. Pour obtenir des résultats plus ciblés, configurez les paramètres du filtre granulaire.

La section Interrupteurs d'intention répertorie les cas dans lesquels le bot est passé de l'intention actuelle à une autre avec les informations suivantes :

- Étape : étape de la conversation au cours de laquelle le bot a changé d'intention.
- Intention convertie en : intention vers laquelle le bot a changé l'intention actuelle.
- Nombre de sessions : nombre de sessions au cours desquelles l'étape et l'intention sont passées en combinaison s'est produite.

> **Note**
>
> Le tableau de bord affiche les 1 000 meilleurs résultats pour un ensemble de paramètres de filtre. Pour obtenir des résultats plus ciblés, configurez les paramètres du filtre granulaire.

Reconnaissance des énoncés

Cette page répertorie tous les énoncés qui ont été manqués ou détectés par votre bot et fournit des outils vous permettant d'ajouter des exemples d'énoncés aux intentions afin de faciliter l'entraînement de votre bot. Voir Énoncés pour une explication de ces définitions. Utilisez les onglets situés en haut pour passer d'une liste d'énoncés manqués à une liste d'énoncés détectés.

> **Note**
>
> Le tableau de bord affiche les 1 000 meilleurs résultats pour un ensemble de paramètres de filtre. Pour obtenir des résultats plus ciblés, configurez les paramètres du filtre granulaire.

Pour ajouter des énoncés à une intention :

1. Cochez la case à côté des énoncés que vous souhaitez ajouter comme exemples d'énoncés pour une intention.
2. Sélectionnez Ajouter à l'intention et choisissez l'intention à laquelle vous souhaitez ajouter les énoncés dans le menu déroulant situé sous Intention.
3. Sélectionnez Ajouter.

Utilisation des API pour les analyses

Cette section décrit les opérations d'API que vous utilisez pour récupérer les analyses d'un bot.

> **Note**
>
> Pour utiliser le ListUtteranceMetricset ListUtteranceAnalyticsData, votre rôle IAM doit disposer des autorisations nécessaires pour effectuer l'ListAggregatedUtterancesopération, ce qui donne accès aux analyses relatives aux énoncés. Consultez Afficher les statistiques relatives aux énoncés pour plus de détails et la politique IAM à appliquer au rôle IAM.

- Les opérations d'API suivantes permettent de récupérer des métriques récapitulatives pour un bot :
  - ListSessionMetrics
  - ListIntentMetrics
  - ListIntentStageMetrics

  - ListUtteranceMetrics
- Les opérations d'API suivantes extraient une liste de métadonnées pour les sessions et les énoncés :
  - ListSessionAnalyticsData
  - ListUtteranceAnalyticsData
- L'ListIntentPathsopération récupère des statistiques relatives à un ordre d'intention suivi par les clients lors de conversations avec un bot.

Filtrage des résultats

Les demandes d'API Analytics nécessitent que vous spécifiiez le `startTime` et`endTime`. L'API renvoie les sessions, les intentions, les étapes d'intention ou les énoncés qui ont commencé après le *`startTime`* et se sont terminés avant le. `endTime`

`filters`est un champ facultatif dans les demandes de l'API Analytics. Il correspond à une liste de AnalyticsSessionFilterAnalyticsIntentFilter, AnalyticsIntentStageFilter, ou AnalyticsUtteranceFilterobjets. Dans chaque objet, utilisez les champs pour créer une expression à filtrer. Par exemple, si vous ajoutez le filtre suivant à la liste, le bot recherche les conversations de plus de 30 secondes.

```
{
    "name": "Duration",
    "operator": "GT",
    "value": "30 sec",
}
```

Récupération de métriques pour un bot

Utilisez les `ListUtteranceMetrics` opérations`ListSessionMetrics`, `ListIntentMetrics``ListIntentStageMetrics`, et pour récupérer les métriques récapitulatives des sessions, des intentions, des étapes d'intention et des énoncés.

Pour ces opérations, renseignez les champs obligatoires suivants :

- Entrez un `startTime` et `endTime` pour définir la plage de temps pour laquelle vous souhaitez récupérer les résultats.
- Spécifiez les mesures que vous souhaitez calculer`metrics`, une liste de AnalyticsSessionMetricAnalyticsIntentMetricAnalyticsIntentStageMetric, ou

d'AnalyticsUtteranceMetricobjets. Dans chaque objet, utilisez le `name` champ pour spécifier la métrique à calculer, le `statistic` champ pour indiquer s'il faut calculer le `SumAverage`, ou le `Max` nombre, et le `order` champ pour indiquer s'il faut trier les résultats par `Descending` ordre `Ascending` ou par ordre.

> **Note**
>
> Les `binBy` objets `metrics` et contiennent tous deux un `order` champ. Vous ne pouvez spécifier le tri `order` que pour l'un des deux objets.

Les autres champs de la demande sont facultatifs. Vous pouvez filtrer et organiser les résultats de différentes manières :

- Filtrage des résultats : utilisez le `filters` champ pour filtrer les résultats. Pour plus d'informations, consultez Filtrage des résultats.
- Regroupement des résultats par catégorie : spécifiez le `groupBy` champ, une liste contenant un seul AnalyticsSessionResultAnalyticsIntentResult, AnalyticsIntentStageResult, ou un AnalyticsUtteranceResultobjet. Dans l'objet, spécifiez le `name` champ contenant la catégorie selon laquelle vous souhaitez regrouper les résultats.

  Si vous spécifiez un `groupBy` champ dans la demande, l'`results`objet de la réponse contient `groupByKeys` une liste de AnalyticsSessionGroupByKey, AnalyticsIntentGroupByKeyAnalyticsIntentStageGroupByKey, ou AnalyticsUtteranceGroupByKeyobjets, chacun avec `name` celui que vous avez spécifié dans la demande et un membre de cette catégorie dans le `value` champ.
- Classification des résultats par heure : spécifiez le `binBy` champ, une liste contenant un seul AnalyticsBinBySpecificationobjet. Dans l'objet, spécifiez le `name` champ contenant `ConversationStartTime` pour classer les résultats en fonction de la date à laquelle la conversation a commencé ou `UtteranceTimestamp` pour classer les résultats en fonction de la date à laquelle l'énoncé a eu lieu. Spécifiez l'intervalle de temps selon lequel vous souhaitez regrouper les résultats dans le `interval` champ, et indiquez si vous souhaitez trier `Ascending` ou l'`Descending`ordre chronologique dans le `order` champ.

  Si vous spécifiez un `binBy` champ dans la demande, l'`results`objet de la réponse contient `binKeys` une liste d'AnalyticsBinKeyobjets, chacun avec `name` celui que vous avez spécifié dans la demande et l'intervalle de temps qui définit ce casier dans le `value` champ.

> **Note**
>
> Les `binBy` objets `metrics` et contiennent tous deux un `order` champ. Vous ne pouvez spécifier le tri `order` que pour l'un des deux objets.

Utilisez les champs suivants pour gérer l'affichage de la réponse :

- Spécifiez un nombre compris entre 1 et 1 000 dans le `maxResults` champ pour limiter le nombre de résultats à renvoyer dans une seule réponse.
- Si le nombre de résultats est supérieur au nombre que vous spécifiez dans le `maxResults` champ, la réponse contient un`nextToken`. Effectuez à nouveau la demande, mais utilisez cette valeur dans le `nextToken` champ pour renvoyer le prochain lot de résultats.

Si vous en utilisez`ListUtteranceMetrics`, vous pouvez spécifier les attributs à renvoyer dans le `attributes` champ. Ce champ correspond à une liste contenant un seul AnalyticsUtteranceAttributeobjet. Spécifiez `LastUsedIntent` dans le `name` champ pour renvoyer l'intention utilisée par Amazon Lex V2 au moment de l'énoncé.

Dans la réponse, le `results` champ correspond à une liste de AnalyticsSessionResult, AnalyticsIntentResultAnalyticsIntentStageResult, ou d'AnalyticsUtteranceResultobjets. Chaque objet contient un `metrics` champ qui renvoie la valeur d'une statistique récapitulative pour une métrique que vous avez demandée, en plus des groupes ou groupes créés à partir des méthodes que vous avez spécifiées.

Récupération des métadonnées pour les sessions et les énoncés dans un bot

Utilisez les ListUtteranceAnalyticsDataopérations ListSessionAnalyticsDataet pour récupérer les métadonnées relatives à des sessions et à des énoncés individuels.

Renseignez les `endTime` champs obligatoires `startTime` et pour définir la plage de temps pour laquelle vous souhaitez récupérer les résultats.

Les autres champs de la demande sont facultatifs. Pour filtrer et trier les résultats :

- Filtrage des résultats : utilisez le `filters` champ pour filtrer les résultats. Pour plus d'informations, consultez Filtrage des résultats.

- Tri des résultats : triez les résultats avec le `sortBy` champ contenant un UtteranceDataSortByobjet SessionDataSortByou. Spécifiez la valeur selon laquelle vous souhaitez trier dans le `name` champ et indiquez s'il faut trier `Ascending` ou `Descending` trier dans le `order` champ.

Utilisez les champs suivants pour gérer l'affichage de la réponse :

- Spécifiez un nombre compris entre 1 et 1 000 dans le `maxResults` champ pour limiter le nombre de résultats à renvoyer dans une seule réponse.
- Si le nombre de résultats est supérieur au nombre que vous spécifiez dans le `maxResults` champ, la réponse contient un`nextToken`. Effectuez à nouveau la demande, mais utilisez cette valeur dans le `nextToken` champ pour renvoyer le prochain lot de résultats.

Dans la réponse, le `utterances` champ `sessions` ou correspond à une liste d'SessionSpecificationUtteranceSpecificationobjets. Chaque objet contient les métadonnées d'une seule session ou d'un seul énoncé.

Récupération des métadonnées pour les sessions et les énoncés dans un bot

Utilisez cette ListIntentPathsopération pour récupérer des statistiques relatives à un ordre d'intention suivi par les clients lors d'une conversation avec un bot.

Pour cette opération, renseignez les champs obligatoires suivants :

- Entrez un `startTime` et `endTime` pour définir la plage de temps pour laquelle vous souhaitez récupérer les résultats.
- Indiquez un `intentPath` pour définir l'ordre d'intention pour lequel vous souhaitez récupérer les métriques. Séparez les intentions du tracé par une barre oblique. Par exemple, renseignez le `intentPath` champ avec **`/BookCar/BookHotel`** pour voir le nombre de fois où les utilisateurs ont invoqué le `BookCar` et `BookHotel` les intentions dans cet ordre.

Utilisez le `filters` champ facultatif pour filtrer les résultats. Pour en savoir plus, consultez Filtrage des résultats.

Afficher les statistiques relatives aux énoncés

Vous pouvez utiliser les statistiques relatives aux énoncés pour déterminer les énoncés que vos utilisateurs envoient à votre bot. Vous pouvez voir à la fois les énoncés qu'Amazon Lex V2 détecte

avec succès et ceux qu'il ne détecte pas. Vous pouvez utiliser ces informations pour optimiser votre bot.

Par exemple, si vous constatez que vos utilisateurs envoient un énoncé indiquant qu'Amazon Lex V2 est absent, vous pouvez ajouter cet énoncé à une intention. La version préliminaire de l'intention est mise à jour avec le nouvel énoncé et vous pouvez la tester avant de la déployer sur votre bot.

Un énoncé est détecté lorsqu'Amazon Lex V2 le reconnaît comme une tentative d'invoquer une intention configurée pour un bot. Un énoncé est oublié lorsqu'Amazon Lex V2 ne le reconnaît pas et l'invoque à sa place. `AMAZON.FallbackIntent`

Les statistiques relatives aux énoncés peuvent être consultées à l'aide de l'`ListUtteranceMetrics`API et de l'`ListAggregatedUtterance`API.

Les statistiques d'énoncé ne sont pas générées à l'aide de l'`ListUtteranceMetrics`API dans les conditions suivantes :

- Le paramètre Child Online Privacy Protection Act était défini sur Oui lorsque le bot a été créé avec la console, ou le `childDirected` champ était défini sur true lorsque le bot a été créé avec l'`CreateBot`opération.

L'`ListUtteranceMetrics`API fournit des fonctionnalités supplémentaires, notamment :

- Plus d'informations sont disponibles, telles que l'intention mappée pour les énoncés détectés.
- Plus de capacité de filtrage (y compris le canal et le mode).
- Plage de dates de conservation plus longue (30 jours).
- Vous pouvez utiliser l'API même si vous avez choisi de ne pas enregistrer les données. La fonctionnalité de console pour les énoncés manqués ou détectés s'appuiera sur `ListUtteranceMetrics` l'API.

Les statistiques d'énoncé ne sont pas générées à l'aide de l'`ListAggregatedUtterance`API dans les conditions suivantes :

- Le paramètre Child Online Privacy Protection Act était défini sur Oui lorsque le bot a été créé avec la console, ou le `childDirected` champ était défini sur true lorsque le bot a été créé avec l'`CreateBot`opération.
- Vous utilisez l'obfuscation des emplacements avec un ou plusieurs emplacements.
- Vous avez choisi de ne pas participer à l'amélioration d'Amazon Lex.

L'`ListAggregatedUtterance`API fournit des fonctionnalités telles que :

- Informations moins détaillées disponibles (aucune intention cartographiée pour les énoncés).
- Capacité de filtrage limitée (sans compter le canal et le mode).
- Période de conservation courte (15 jours).

À l'aide des statistiques relatives aux énoncés, vous pouvez savoir si un énoncé spécifique a été détecté ou oublié, ainsi que la dernière fois que cet énoncé a été utilisé lors d'une interaction avec un bot.

Amazon Lex V2 stocke les énoncés en continu pendant que les utilisateurs interagissent avec votre bot. Vous pouvez consulter les statistiques à l'aide de la console ou de l'`ListAggregatedUtterances`opération. Il a une durée de conservation des données de 15 jours et n'est pas disponible si l'utilisateur a choisi de ne pas le stocker. Vous pouvez supprimer des énoncés à l'aide de cette `DeleteUtterances` opération ou en refusant le stockage des données. Tous les énoncés sont supprimés si vous fermez votre AWS compte. Les énoncés enregistrés sont chiffrés à l'aide d'une clé gérée par le serveur.

Lorsque vous supprimez une version de bot, les statistiques d'énoncé sont disponibles pour cette version pendant 30 jours au maximum et pendant 15 jours en `ListUtteranceMetrics` cas d'utilisation. `ListAggregatedUtterances` Vous ne pouvez pas consulter les statistiques relatives à la version supprimée dans la console Amazon Lex V2. Pour consulter les statistiques des versions supprimées, vous pouvez utiliser à la fois les `ListUtteranceMetrics` opérations `ListAggregatedUtterances` et les opérations.

Avec les `ListUtteranceMetrics` API `ListAggregatedUtterances` et, les énoncés sont agrégés en fonction du texte de l'énoncé. Par exemple, tous les cas où le client a utilisé l'expression « Je veux commander une pizza » sont regroupés sur la même ligne dans une réponse. Lorsque vous utilisez l'RecognizeUtteranceopération, le texte utilisé est le transcrit d'entrée.

Pour utiliser les `ListUtteranceMetrics` API `ListAggregatedUtterances` et, appliquez la politique suivante à un rôle.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ListAggregatedUtterancesPolicy",
```

```
            "Effect": "Allow",
            "Action": "lex:ListAggregatedUtterances",
            "Resource": "*"
        }
    ]
}
```

Gestion des autorisations d'accès pour les analyses

Pour permettre à un utilisateur d'accéder aux analyses, associez une politique à un rôle IAM qui permet au rôle d'appeler les opérations d'API à des fins d'analyse. Vous pouvez associer le rôle Politique gérée par AWS : AmazonLexFullAccess au rôle IAM pour fournir un accès complet aux opérations de l'API Amazon Lex, ou vous pouvez créer une politique personnalisée autorisant uniquement les autorisations d'analyse et l'associer à un rôle IAM.

Pour créer une politique personnalisée contenant des autorisations pour les analyses

1. Si vous devez d'abord créer un rôle IAM, suivez les étapes de la section Création d'un rôle pour déléguer des autorisations à un utilisateur IAM.
2. Suivez les étapes de la section Création de politiques IAM pour créer une politique à l'aide de l'objet JSON suivant. Pour permettre l'accès analytique à des robots spécifiques pour le rôle IAM, ajoutez l'ARN de chaque bot `Resource` dans le champ. Remplacez la *`région`*, l'*`identifiant du compte`* et le *`BOTID`* par les valeurs correspondant aux robots. Vous pouvez également remplacer l'identifiant du relevé par le nom de votre choix. *`AnalyticsActions`*

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Sid": "AnalyticsActions",
               "Effect": "Allow",
               "Action": [
                   "lex:ListAggregatedUtterances",
                   "lex:ListIntentMetrics",
                   "lex:ListSessionAnalyticsData",
                   "lex:ListIntentPaths",
                   "lex:ListIntentStageMetrics",
                   "lex:ListSessionMetrics"
               ],
               "Resource": [
                   "arn:aws:lex:region:account-id:bot/BOTID"
   ```

```
            ]
        }
    ]
}
```

3. Associez la politique que vous avez créée au rôle auquel vous souhaitez accorder des autorisations d'analyse en suivant les étapes de la section Ajouter et supprimer des autorisations d'identité IAM.
4. Le rôle doit désormais être autorisé à consulter les analyses des robots que vous avez spécifiés.

Activation des journaux de conversation

Utilisez les journaux de conversation pour enregistrer les conversations des utilisateurs avec votre bot. Consultez ces journaux pour identifier les problèmes liés aux interactions de votre bot avec les utilisateurs et modifiez le comportement de votre bot grâce à ces informations. Cette section décrit également comment masquer les valeurs des emplacements afin de protéger la confidentialité des utilisateurs.

Rubriques

- Journalisation à l'aide des journaux de conversation
- Masquer les valeurs des créneaux dans les journaux de conversation
- Capture sélective du journal des conversations

Journalisation à l'aide des journaux de conversation

Vous activez les journaux de conversation pour stocker les interactions de bot. Vous pouvez utiliser ces journaux pour vérifier les performances de votre bot et résoudre les problèmes liés aux conversations. Vous pouvez enregistrer le texte de l'RecognizeTextopération. Vous pouvez enregistrer à la fois du texte et du son pour l'RecognizeUtteranceopération. En activant les journaux de conversation, vous obtenez une vue détaillée des conversations que les utilisateurs ont avec votre bot.

Par exemple, une session avec votre bot a un ID de session. Vous pouvez utiliser cet ID pour obtenir la transcription de la conversation, y compris les déclarations de l'utilisateur et les réponses correspondantes du bot. Vous obtenez également des métadonnées telles que le nom d'intention et les valeurs d'emplacement pour un énoncé.

> **Note**
>
> Vous ne pouvez pas utiliser les journaux de conversation avec un bot soumis à la loi Children's Online Privacy Protection Act (COPPA).

Les journaux de conversation sont configurés pour un alias. Chaque alias peut avoir des paramètres différents pour les journaux de texte et d'audio. Vous pouvez activer les journaux de texte, les journaux d'audio ou les deux pour chaque alias. Les journaux de texte stockent les entrées de texte, les transcriptions des entrées audio et les métadonnées associées dans les CloudWatch journaux. Les journaux audio stockent les entrées audio dans Amazon S3. Vous pouvez activer le chiffrement des journaux texte et audio à l'aide de clés CMK gérées par le client AWS KMS .

Pour configurer la journalisation, utilisez la console CreateBotAliasou l'UpdateBotAliasopération or. Après avoir activé les journaux de conversation pour un alias, l'RecognizeUtteranceopération RecognizeTextou pour cet alias enregistre le texte ou les énoncés audio dans le groupe de CloudWatch journaux Logs ou le compartiment S3 configuré.

Rubriques

- Politiques IAM pour les journaux de conversation
- Configuration des journaux de conversation
- Afficher les journaux textuels dans Amazon CloudWatch Logs
- Accès aux journaux audio dans Amazon S3
- Surveillance de l'état du journal des conversations à l'aide de CloudWatch métriques

Politiques IAM pour les journaux de conversation

Selon le type de journalisation que vous sélectionnez, Amazon Lex V2 nécessite l'autorisation d'utiliser les compartiments Amazon CloudWatch Logs et Amazon Simple Storage Service (S3) pour stocker vos journaux. Vous devez créer des AWS Identity and Access Management rôles et des autorisations pour permettre à Amazon Lex V2 d'accéder à ces ressources.

Création d'un rôle IAM et de stratégies pour les journaux de conversation

Pour activer les journaux de conversation, vous devez accorder une autorisation d'écriture à CloudWatch Logs et à Amazon S3. Si vous activez le chiffrement d'objets pour vos objets S3, vous devez accorder l'autorisation d'accès aux AWS KMS clés utilisées pour chiffrer les objets.

Vous pouvez utiliser la console IAM, l'API IAM ou le AWS Command Line Interface pour créer le rôle et les politiques. Ces instructions utilisent le AWS CLI pour créer le rôle et les politiques.

> **Note**
>
> Le code suivant est formaté pour Linux et macOS. Sous Windows, remplacez le caractère de continuité de ligne Linux (\) par le caret (^).

Pour créer un rôle IAM pour les journaux de conversation

1. Créez un document dans le répertoire courant appelé **`LexConversationLogsAssumeRolePolicyDocument.json`**, ajoutez-y le code suivant et enregistrez-le. Ce document de politique ajoute Amazon Lex V2 en tant qu'entité de confiance au rôle. Cela permet à Amazon Lex d'assumer le rôle de fournir des journaux aux ressources configurées pour les journaux de conversation.

   ```
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
           "Service": "lexv2.amazonaws.com"
         },
         "Action": "sts:AssumeRole"
       }
     ]
   }
   ```

2. Dans le AWS CLI, exécutez la commande suivante pour créer le rôle IAM pour les journaux de conversation.

   ```
   aws iam create-role \
       --role-name role-name \
       --assume-role-policy-document file://
   LexConversationLogsAssumeRolePolicyDocument.json
   ```

Ensuite, créez et associez une politique au rôle qui permet à Amazon Lex V2 d'écrire dans CloudWatch Logs.

Pour créer une politique IAM pour enregistrer le texte d'une conversation dans Logs CloudWatch

1. Créez un document dans le répertoire actuel appelé**`LexConversationLogsCloudWatchLogsPolicy.json`**, ajoutez-y la politique IAM suivante et enregistrez-le.

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [
              "logs:CreateLogStream",
              "logs:PutLogEvents"
          ],
          "Resource": "arn:aws:logs:region:account-id:log-group:log-group-name:*"
      }
  ]
}
```

2. Dans le AWS CLI, créez la politique IAM qui accorde l'autorisation d'écriture au groupe de CloudWatch journaux Logs.

```
aws iam create-policy \
    --policy-name cloudwatch-policy-name \
    --policy-document file://LexConversationLogsCloudWatchLogsPolicy.json
```

3. Associez la politique au rôle IAM que vous avez créé pour les journaux de conversation.

```
aws iam attach-role-policy \
    --policy-arn arn:aws:iam::account-id:policy/cloudwatch-policy-name \
    --role-name role-name
```

Si vous enregistrez du son dans un compartiment S3, créez une politique permettant à Amazon Lex V2 d'écrire dans le compartiment.

Pour créer une politique IAM pour la journalisation audio dans un compartiment S3

1. Créez un document dans le répertoire courant appelé **`LexConversationLogsS3Policy.json`**, ajoutez la stratégie suivante et enregistrez-le.

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [
              "s3:PutObject"
          ],
          "Resource": "arn:aws:s3:::bucket-name/*"
      }
  ]
}
```

2. Dans le AWS CLI, créez la politique IAM qui accorde l'autorisation d'écriture à votre compartiment S3.

```
aws iam create-policy \
    --policy-name s3-policy-name \
    --policy-document file://LexConversationLogsS3Policy.json
```

3. Attachez la stratégie au rôle que vous avez créé pour les journaux de conversation.

```
aws iam attach-role-policy \
    --policy-arn arn:aws:iam::account-id:policy/s3-policy-name \
    --role-name role-name
```

Octroi de l'autorisation de transmettre un rôle IAM

Lorsque vous utilisez la console AWS Command Line Interface, le ou un AWS SDK pour spécifier un rôle IAM à utiliser pour les journaux de conversation, l'utilisateur qui spécifie le rôle IAM des journaux de conversation doit être autorisé à transmettre le rôle à Amazon Lex V2. Pour permettre à l'utilisateur de transmettre le rôle à Amazon Lex V2, vous devez accorder l'`PassRole`autorisation à l'utilisateur, au rôle ou au groupe IAM de l'utilisateur.

La stratégie suivante définit l'autorisation d'accorder à l'utilisateur, au rôle ou au groupe. Vous pouvez utiliser les clés de condition `iam:PassedToService` et `iam:AssociatedResourceArn` pour

limiter la portée de l'autorisation. Pour plus d'informations, consultez la section Octroi à un utilisateur des autorisations pour transmettre un rôle à un AWS service et les clés contextuelles IAM et AWS STS Condition dans le guide de l'AWS Identity and Access Management utilisateur.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::account-id:role/role-name",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "lexv2.amazonaws.com"
                },
                "StringLike": {
                    "iam:AssociatedResourceARN": "arn:aws:lex:region:account-
id:bot:bot-name:bot-alias"
                }
            }
        }
    ]
}
```

Configuration des journaux de conversation

Vous activez et désactivez les journaux de conversation à l'aide de la console ou du `conversationLogSettings` champ de l'`UpdateBotAlias`opération `CreateBotAlias` or. Vous pouvez activer ou désactiver les journaux audio, les journaux de texte ou les deux. La journalisation démarre sur les nouvelles sessions de bot. Les modifications apportées aux paramètres du journal ne sont pas prises en compte pour les sessions actives.

Pour stocker des journaux de texte, utilisez un groupe de CloudWatch journaux Amazon Logs dans votre AWS compte. Vous pouvez utiliser n'importe quel groupe de journaux valide. Le groupe de journaux doit se trouver dans la même région que le bot Amazon Lex V2. Pour plus d'informations sur la création d'un groupe de CloudWatch journaux, consultez la section Working with Log Groups and Log Streams dans le guide de l'utilisateur Amazon CloudWatch Logs.

Pour stocker des journaux audio, utilisez un compartiment Amazon S3 dans votre AWS compte. Vous pouvez utiliser n'importe quel compartiment S3 valide. Le compartiment doit se trouver dans la même région que le bot Amazon Lex V2. Pour plus d'informations sur la création d'un compartiment

S3, consultez la section Création d'un compartiment dans le guide de démarrage Amazon Simple Storage Service.

Lorsque vous gérez les journaux de conversation à l'aide de la console, celle-ci met à jour votre rôle de service afin qu'elle ait accès au groupe de journaux et au compartiment S3.

Si vous n'utilisez pas la console, vous devez fournir un rôle IAM avec des politiques qui permettent à Amazon Lex V2 d'écrire dans le groupe de journaux ou le compartiment configuré. Si vous créez un rôle lié à un service à l'aide du AWS Command Line Interface, vous devez ajouter un suffixe personnalisé au rôle à l'aide de l'`custom-suffix`option, comme dans l'exemple suivant. Pour plus d'informations, consultez Création d'un rôle IAM et de stratégies pour les journaux de conversation.

```
aws iam create-service-linked-role \
    --aws-service-name lexv2.amazon.aws.com \
    --custom-suffix suffix
```

Le rôle IAM que vous utilisez pour activer les journaux de conversation doit disposer de cette `iam:PassRole` autorisation. La politique suivante doit être associée au rôle :

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::account:role/role"
        }
    ]
}
```

Activation des journaux de conversation

Pour activer les journaux à l'aide de la console

1. Ouvrez la console Amazon Lex V2 https://console.aws.amazon.com/lexv2.
2. Dans la liste, choisissez un bot.
3. Dans le menu de gauche, sélectionnez Alias.
4. Dans la liste des alias, choisissez l'alias pour lequel vous souhaitez configurer les journaux de conversation.

5. Dans la section Journaux de conversation, choisissez Gérer les journaux de conversation.
6. Pour les journaux de texte, choisissez Activer, puis entrez le nom du groupe de CloudWatch journaux Amazon Logs.
7. Pour les journaux audio, choisissez Activer, puis entrez les informations du compartiment S3.
8. Facultatif. Pour chiffrer les journaux audio, choisissez la AWS KMS clé à utiliser pour le chiffrement.
9. Choisissez Save (Enregistrer) pour démarrer la journalisation des conversations. Si nécessaire, Amazon Lex V2 mettra à jour votre rôle de service avec les autorisations d'accès au groupe de CloudWatch journaux Logs et au compartiment S3 sélectionné.

Désactivation des journaux de conversation

Pour désactiver les journaux à l'aide de la console

1. Ouvrez la console Amazon Lex V2 https://console.aws.amazon.com/lexv2.
2. Dans la liste, choisissez un bot.
3. Dans le menu de gauche, sélectionnez Alias.
4. Dans la liste des alias, choisissez l'alias pour lequel vous souhaitez configurer les journaux de conversation.
5. Dans la section Journaux de conversation, choisissez Gérer les journaux de conversation.
6. Désactivez l'enregistrement de texte, l'enregistrement audio ou les deux pour désactiver l'enregistrement.
7. Choisissez Save (Enregistrer) pour arrêter la journalisation des conversations.

Afficher les journaux textuels dans Amazon CloudWatch Logs

Amazon Lex V2 stocke les journaux de texte de vos conversations dans Amazon CloudWatch Logs. Pour consulter les journaux, utilisez la console ou l'API CloudWatch Logs. Pour plus d'informations, consultez les données du journal de recherche à l'aide de modèles de filtres et la syntaxe de requête CloudWatch Logs Insights dans le guide de l'utilisateur Amazon CloudWatch Logs.

Pour consulter les journaux à l'aide de la console Amazon Lex V2

1. Ouvrez la console Amazon Lex V2 https://console.aws.amazon.com/lexv2.
2. Dans la liste, choisissez un bot.

3. Dans le menu de gauche, choisissez Analytics, puis sélectionnez CloudWatch Metrics.
4. Consultez les statistiques de votre bot sur la page CloudWatch des statistiques.

Vous pouvez également utiliser la CloudWatch console ou l'API pour consulter les entrées de votre journal. Pour rechercher des entrées de journal, accédez au groupe de journaux que vous avez configuré pour l'alias. Vous pouvez trouver le préfixe du flux de journal pour vos journaux dans la console Amazon Lex V2 ou en utilisant l'DescribeBotAliasopération.

Les entrées de journal correspondant à un énoncé d'utilisateur se trouvent dans plusieurs flux de journaux. Un énoncé dans la conversation comporte une entrée dans l'un des flux de journaux avec le préfixe spécifié. Une entrée du flux de journal contient les informations suivantes :

version du message

Version du schéma du message.

bot

Informations sur le bot avec lequel le client interagit.

messages

La réponse que le bot a renvoyée à l'utilisateur.

Contexte de l'énoncé

Informations sur le traitement de cet énoncé.

- `runtimeHints`—contexte d'exécution utilisé pour transcrire et interpréter les entrées de l'utilisateur. Pour plus d'informations, consultez Amélioration de la reconnaissance des valeurs des créneaux grâce à des indices d'exécution.
- `slotElicitationStyle`—Style d'élicitation de créneaux utilisé pour interpréter les entrées de l'utilisateur. Pour plus d'informations, consultez Capture de valeurs de créneaux avec des styles d'orthographe.

État de la session

État actuel de la conversation entre l'utilisateur et le bot. Pour plus d'informations, consultez Gestion des conversations.

interprétations

Une liste d'intentions déterminées par Amazon Lex V2 pourrait satisfaire l'énoncé de l'utilisateur. Utilisation des scores de confiance.

Source d'interprétation

Indique si un emplacement est résolu par Amazon Lex ou Amazon Bedrock. Valeurs : Lex | Bedrock

sessionId

Identifiant de la session utilisateur en cours de conversation.

inputTranscript

Une transcription de l'entrée de l'utilisateur.

- Pour la saisie de texte, il s'agit du texte saisi par l'utilisateur. Pour l'entrée DTMF, il s'agit de la clé saisie par l'utilisateur.
- Pour la saisie vocale, il s'agit du texte en lequel Amazon Lex V2 convertit l'énoncé de l'utilisateur afin d'invoquer une intention ou de remplir un espace.

rawInputTranscript

Transcription brute des données saisies par l'utilisateur avant tout traitement de texte. Remarque : Le traitement de texte concerne uniquement les paramètres régionaux en-US et en-GB.

transcriptions

Une liste de transcriptions potentielles des entrées de l'utilisateur. Pour plus d'informations, consultez Utilisation des scores de fiabilité de la transcription vocale.

Transcription brute

Utilisation des scores de confiance de la transcription vocale. Pour plus d'informations, consultez Utilisation des scores de fiabilité de la transcription vocale.

Exposé manqué

Indique si Amazon Lex V2 a pu reconnaître l'énoncé de l'utilisateur.

requestId

Amazon Lex V2 a généré un ID de demande pour la saisie par l'utilisateur.

timestamp

Horodatage de l'entrée de l'utilisateur.

Developer Override

Indique si le flux de conversation a été mis à jour à l'aide d'un crochet de dialogue. Pour plus d'informations sur l'utilisation d'un crochet de code de dialogue, consultezActivation d'une logique personnalisée avec des AWS Lambda fonctions.

Mode d'entrée

Indique le type d'entrée. Cela peut être audio, DTMF ou texte.

requestAttributes

Les attributs de demande utilisés lors du traitement de l'entrée de l'utilisateur.

Propriétés audio

Si les journaux de conversation audio sont activés et que l'entrée utilisateur était au format audio, inclut la durée totale de l'entrée audio, la durée de la voix et la durée du silence dans l'audio. Il inclut également un lien vers le fichier audio.

BargeIn

Indique si la saisie de l'utilisateur a interrompu la réponse précédente du bot.

Motif de la réponse

La raison pour laquelle une réponse a été générée. Les valeurs suivantes sont possibles :

- `UtteranceResponse`— réponse aux entrées de l'utilisateur
- `StartTimeout`— réponse générée par le serveur lorsque l'utilisateur n'a pas fourni d'entrée
- `StillWaitingResponse`— réponse générée par le serveur lorsque l'utilisateur demande au bot d'attendre
- `FulfillmentInitiated`— réponse générée par le serveur indiquant que le traitement des commandes est sur le point d'être lancé
- `FulfillmentStartedResponse`— réponse générée par le serveur indiquant que l'expédition a commencé
- `FulfillmentUpdateResponse`— réponse périodique générée par le serveur pendant que l'expédition est en cours
- `FulfillmentCompletedResponse`— réponse générée par le serveur lorsque l'exécution est terminée.

operationName

L'API utilisée pour interagir avec le bot. Peut être l'un des `PutSessionRecognizeText`,`RecognizeUtterance`, ou`StartConversation`.

```
{
    "message-version": "2.0",
    "bot": {
        "id": "string",
        "name": "string",
        "aliasId": "string",
        "aliasName": "string",
        "localeId": "string",
        "version": "string"
    },
    "messages": [
        {
            "contentType": "PlainText | SSML | CustomPayload | ImageResponseCard",
            "content": "string",
            "imageResponseCard": {
                "title": "string",
                "subtitle": "string",
                "imageUrl": "string",
                "buttonsList": [
                    {
                        "text": "string",
                        "value": "string"
                    }
                ]
            }
        }
    ],
    "utteranceContext": {
        "activeRuntimeHints": {
            "slotHints": {
                "string": {
                    "string": {
                        "runtimeHintValues": [
                            {
                                "phrase": "string"
                            },
```

```
                        {
                            "phrase": "string"
                        }
                    ]
                }
            }
        },
        "slotElicitationStyle": "string"
    },
    "sessionState": {
        "dialogAction": {
            "type": "Close | ConfirmIntent | Delegate | ElicitIntent | ElicitSlot",
            "slotToElicit": "string"
        },
        "intent": {
            "name": "string",
            "slots": {
                "string": {
                    "value": {
                       "interpretedValue": "string",
                       "originalValue": "string",
                       "resolvedValues": [ "string" ]
                    }
                 },
                "string": {
                    "shape": "List",
                    "value": {
                        "originalValue": "string",
                        "interpretedValue": "string",
                        "resolvedValues": [ "string" ]
                    },
                    "values": [
                        {
                            "shape": "Scalar",
                            "value": {
                                "originalValue": "string",
                                "interpretedValue": "string",
                                "resolvedValues": [ "string" ]
                            }
                        },
                        {
                            "shape": "Scalar",
                            "value": {
```

```
                                "originalValue": "string",
                                "interpretedValue": "string",
                                "resolvedValues": [ "string" ]
                            }
                        }
                    ]
                }
            },
            "kendraResponse": {
                // Only present when intent is KendraSearchIntent. For details, see
                // https://docs.aws.amazon.com/kendra/latest/dg/
API_Query.html#API_Query_ResponseSyntax
                },
            "state": "InProgress | ReadyForFulfillment | Fulfilled | Failed",
            "confirmationState": "Confirmed | Denied | None"
        },
        "originatingRequestId": "string",
        "sessionAttributes": {
            "string": "string"
        },
        "runtimeHints": {
            "slotHints": {
                "string": {
                    "string": {
                        "runtimeHintValues": [
                            {
                                "phrase": "string"
                            },
                            {
                                "phrase": "string"
                            }
                        ]
                    }
                }
            }
        }
    },
   "dialogEventLogs": [
        {
   // only for conditional
     "conditionalEvaluationResult":[
      // all the branches until true

     {
```

```
    "conditionalBranchName": "string",
    "expressionString": "string",
    "evaluatedExpression": "string",
    "evaluationResult": "true | false"
    }
   ],
  "dialogCodeHookInvocationLabel": "string",
  "response": "string",
  "nextStep": {
       "dialogAction": {
           "type": "Close | ConfirmIntent | Delegate | ElicitIntent | ElicitSlot",
           "slotToElicit": "string"
       },
      "intent": {
                        "name": "string",
          "slots": {
              }
       }
      }
   ]
   "interpretations": [
       {
           "interpretationSource": "Bedrock | Lex",
           "nluConfidence": "string",
           "intent": {
               "name": "string",
               "slots": {
                   "string": {
                       "value": {
                           "originalValue": "string",
                           "interpretedValue": "string",
                           "resolvedValues": [ "string" ]
                       }
                   },
                   "string": {
                       "shape": "List",
                       "value": {
                           "interpretedValue": "string",
                           "originalValue": "string",
                           "resolvedValues": [ "string" ]
                       },
                       "values": [
                           {
                               "shape": "Scalar",
```

```
                                "value": {
                                    "interpretedValue": "string",
                                    "originalValue": "string",
                                    "resolvedValues": [ "string" ]
                                }
                            },
                            {
                                "shape": "Scalar",
                                "value": {
                                    "interpretedValue": "string",
                                    "originalValue": "string",
                                    "resolvedValues": [ "string" ]
                                }

                            }
                        ]
                    }
                },
                "kendraResponse": {
                    // Only present when intent is KendraSearchIntent. For details, see

                    // https://docs.aws.amazon.com/kendra/latest/dg/
API_Query.html#API_Query_ResponseSyntax
                    },
                "state": "InProgress | ReadyForFulfillment | Fulfilled | Failed",
                "confirmationState": "Confirmed | Denied | None"
                },
            "sentimentResponse": {
                "sentiment": "string",
                "sentimentScore": {
                    "positive": "string",
                    "negative": "string",
                    "neutral": "string",
                    "mixed": "string"
                }
            }
        }
    ],
    "sessionId": "string",
    "inputTranscript": "string",
    "rawInputTranscript": "string",
    "transcriptions": [
        {
            "transcription": "string",
```

```
            "rawTranscription": "string",
            "transcriptionConfidence": "number",
            },
            "resolvedContext": {
                "intent": "string"
            },
            "resolvedSlots": {
                "string": {
                    "name": "slotName",
                    "shape": "List",
                    "value": {
                        "originalValue": "string",
                        "resolvedValues": [
                            "string"
                        ]
                    }
                }
            }
        }
    ],
    "missedUtterance": "bool",
    "requestId": "string",
    "timestamp": "string",
    "developerOverride": "bool",
    "inputMode": "DTMF | Speech | Text",
    "requestAttributes": {
        "string": "string"
    },
    "audioProperties": {
        "contentType": "string",
        "s3Path": "string",
        "duration": {
            "total": "integer",
            "voice": "integer",
            "silence": "integer"
        }
    },
    "bargeIn": "string",
    "responseReason": "string",
    "operationName": "string"
}
```

Le contenu de l'entrée du journal dépend du résultat d'une transaction et de la configuration du bot et de la demande.

- Les champs `intent`, `slots` et `slotToElicit` n'apparaissent pas dans une entrée si le champ `missedUtterance` a la valeur `true`.
- Le champ `s3PathForAudio` n'apparaît pas si les journaux audio sont désactivés ou si le champ `inputDialogMode` est `Text`.
- Le champ `responseCard` n'apparaît que lorsque vous avez défini une carte de réponse pour le bot.
- La carte `requestAttributes` n'apparaît que si vous avez spécifié des attributs de demande dans la demande.
- Le `kendraResponse` champ n'est présent que lorsqu'il `AMAZON.KendraSearchIntent` fait une demande de recherche dans un index Amazon Kendra.
- Le `developerOverride` champ est vrai lorsqu'une intention alternative a été spécifiée dans la fonction Lambda du bot.
- La carte `sessionAttributes` n'apparaît que si vous avez spécifié des attributs de session dans la demande.
- La carte `sentimentResponse` n'apparaît que si vous configurez le bot pour qu'il renvoie des valeurs de sentiment.

> **Note**
>
> Le format d'entrée peut changer sans modification correspondante dans `messageVersion`. Le code ne devrait pas générer une erreur si de nouveaux champs sont présents.

Accès aux journaux audio dans Amazon S3

Amazon Lex V2 stocke les journaux audio de vos conversations dans un compartiment S3.

Vous pouvez utiliser la console ou l'API Amazon S3 pour accéder aux journaux audio. Vous pouvez voir le préfixe de clé d'objet S3 des fichiers audio dans la console Amazon Lex V2 ou dans le `conversationLogSettings` champ de la réponse à l'`DescribeBotAlias`opération.

Surveillance de l'état du journal des conversations à l'aide de CloudWatch métriques

Utilisez Amazon CloudWatch pour surveiller les statistiques de livraison de vos journaux de conversations. Vous pouvez définir des alarmes sur les métriques de manière à être informé des éventuels problèmes liés à la journalisation.

Amazon Lex V2 fournit quatre métriques dans l'espace de `AWS/Lex` noms pour les journaux de conversation :

- `ConversationLogsAudioDeliverySuccess`
- `ConversationLogsAudioDeliveryFailure`
- `ConversationLogsTextDeliverySuccess`
- `ConversationLogsTextDeliveryFailure`

Les indicateurs de réussite indiquent qu'Amazon Lex V2 a correctement enregistré vos journaux audio ou texte vers leur destination.

Les indicateurs d'échec indiquent qu'Amazon Lex V2 n'a pas pu envoyer les journaux audio ou textuels à la destination spécifiée. Généralement, il s'agit d'une erreur de configuration. Lorsque vos métriques d’échec sont supérieures à zéro, vérifiez les points suivants :

- Assurez-vous qu'Amazon Lex V2 est une entité fiable pour le rôle IAM.
- Pour l'enregistrement de texte, assurez-vous que le groupe de CloudWatch journaux des journaux existe. Pour la journalisation audio, assurez-vous que le compartiment S3 existe.
- Assurez-vous que le rôle IAM utilisé par Amazon Lex V2 pour accéder au groupe de CloudWatch journaux ou au compartiment S3 dispose d'une autorisation d'écriture pour le groupe de journaux ou le compartiment.
- Assurez-vous que le compartiment S3 existe dans la même région que le bot Amazon Lex V2 et qu'il appartient à votre compte.

Masquer les valeurs des créneaux dans les journaux de conversation

Amazon Lex V2 vous permet de masquer ou de masquer le contenu des emplacements afin qu'il ne soit pas visible. Pour protéger les données sensibles capturées comme valeurs d’emplacement, vous pouvez activer l'obfuscation d'emplacement pour masquer ces valeurs lors de la journalisation.

Lorsque vous choisissez de masquer les valeurs d'emplacement, Amazon Lex V2 remplace la valeur de l'emplacement par le nom de l'emplacement dans les journaux de conversation. Pour un emplacement appelé `full_name`, la valeur de l'emplacement sera obfusquée comme suit :

```
Before:
    My name is John Stiles
After:
    My name is {full_name}
```

Si un énoncé contient des crochets ({}) Amazon Lex V2 les remplace par deux barres obliques (\ \). Par exemple, le texte `{John Stiles}` est obfusqué comme suit :

```
Before:
    My name is {John Stiles}
After:
    My name is \\{{full_name}\\}
```

Les valeurs d'emplacement sont obfusquées dans les journaux de conversation. Les valeurs des créneaux sont toujours disponibles dans la réponse des `RecognizeUtterance` opérations `RecognizeText` et, et les valeurs des créneaux sont disponibles pour vos fonctions Lambda de validation et d'exécution. Si vous utilisez des valeurs d'emplacement dans vos invites ou vos réponses, ces valeurs ne sont pas obfusquées dans les journaux de conversation.

Au premier tour d'une conversation, Amazon Lex V2 masque les valeurs des créneaux s'il reconnaît un créneau et une valeur de créneau dans l'énoncé. Si aucune valeur de slot n'est reconnue, Amazon Lex V2 ne masque pas l'énoncé.

Au deuxième tour et aux tours suivants, Amazon Lex V2 sait quel emplacement doit être sélectionné et si la valeur de l'emplacement doit être masquée. Si Amazon Lex V2 reconnaît la valeur de l'emplacement, celle-ci est masquée. Si Amazon Lex V2 ne reconnaît pas une valeur, l'énoncé entier est masqué. Les valeurs d'emplacement dans les énoncés manqués ne sont pas obfusqués.

Amazon Lex V2 ne masque pas non plus les valeurs d'emplacement que vous stockez dans les attributs de demande ou de session. Si vous stockez des valeurs d'emplacement qui doivent être masquées en tant qu'attribut, vous devez chiffrer ou obfusquer la valeur.

Amazon Lex V2 ne masque pas la valeur du slot dans le son. Il obfusque la valeur de l’emplacement dans la transcription audio.

Vous pouvez choisir les emplacements à masquer à l'aide de la console ou de l'API Amazon Lex V2. Dans la console, choisissez Slot obfuscation (Obfuscation d'emplacement) dans les paramètres d'un emplacement. Si vous utilisez l'API, définissez le `obfuscationSetting` champ du slot sur `DEFAULT_OBFUSCATION` lorsque vous appelez l'UpdateSlotopération CreateSlotor.

Capture sélective du journal des conversations

La capture sélective des journaux de conversation permet à l'utilisateur de sélectionner la manière dont les journaux de conversation sont capturés avec le texte et les données audio des conversations en direct.

Pour activer et capturer le résultat de la fonctionnalité de capture sélective des journaux de conversation, vous devez activer la fonctionnalité dans la console Amazon Lex V2 et activer les attributs de session requis dans les paramètres de l'API afin de capturer la sortie sélectionnée à partir des journaux.

Vous pouvez sélectionner les options suivantes pour la capture sélective du journal des conversations :

- texte uniquement
- audio uniquement
- texte et audio

Vous pouvez capturer des parties spécifiques de la conversation et choisir si le son, le texte ou les deux sont capturés pour le journal des conversations.

> **Note**
>
> La capture sélective des journaux de conversation fonctionne uniquement pour Amazon Lex V2.

Rubriques

- Gérer la capture sélective des journaux de conversation
- Exemple de capture sélective du journal des conversations

Gérer la capture sélective des journaux de conversation

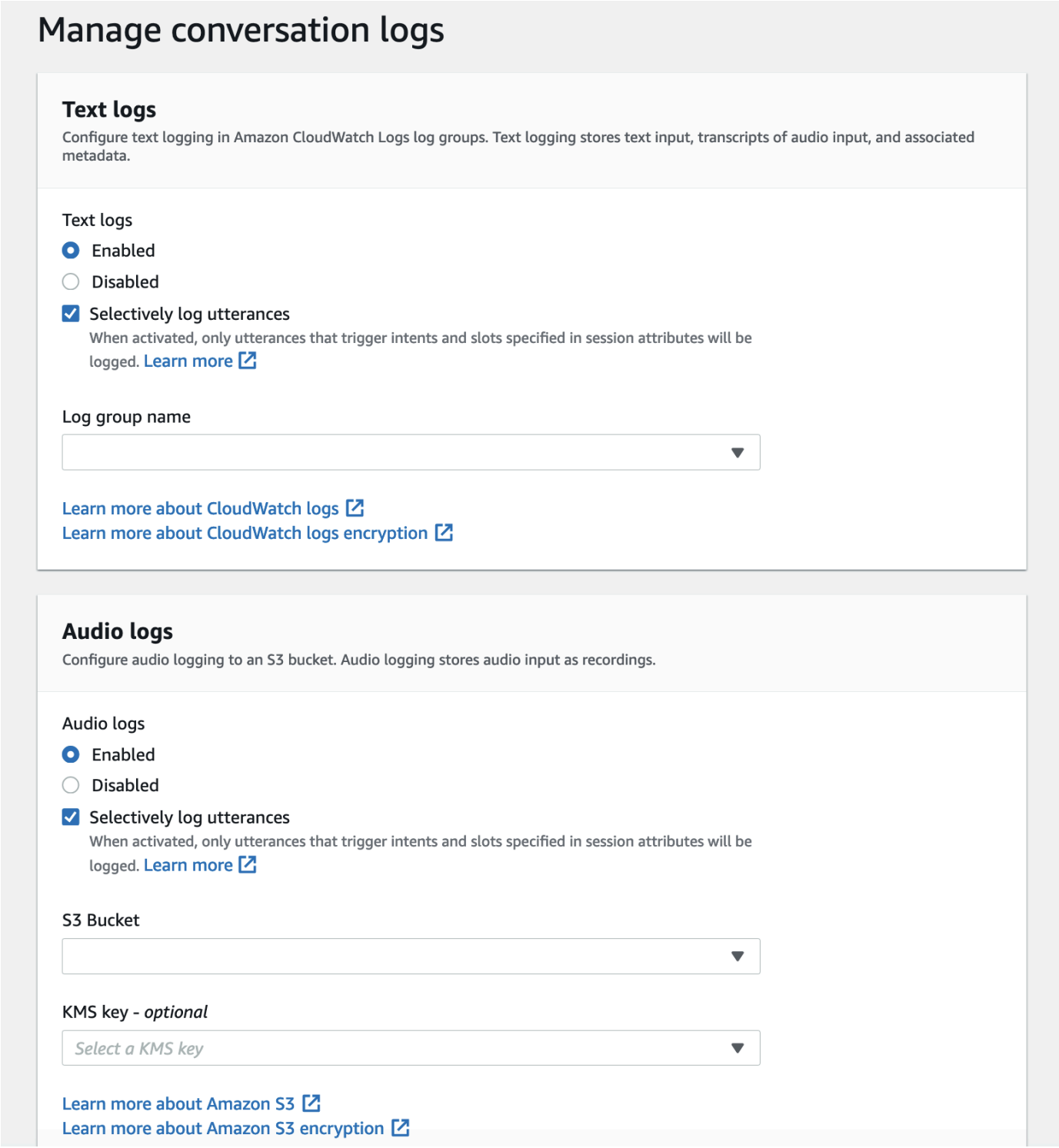
À l'aide de la console Lex, vous pouvez activer les paramètres de capture sélective des journaux de conversation et choisir les emplacements pour lesquels vous souhaitez activer la capture sélective des journaux de conversation.

Activez la capture sélective du journal des conversations dans la console Amazon Lex V2 :

1. Connectez-vous à la console Amazon Lex V2 AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/lexv2/home.
2. Sélectionnez Bots dans les panneaux de gauche et choisissez le bot pour lequel vous souhaitez activer la capture sélective du journal des conversations. Utilisez un bot existant ou créez-en un nouveau.
3. Choisissez les alias pour le bot que vous avez sélectionné dans la section Déploiement sur le panneau de gauche.
4. Choisissez l'alias de votre bot, puis sélectionnez Gérer les journaux de conversation.
5. Dans le panneau Gérer les journaux de conversation, pour les journaux de texte, choisissez si les journaux de texte sont activés ou désactivés en sélectionnant le bouton radio. Si vous choisissez Activé pour les journaux de texte, vous devez saisir un nom de groupe de journaux ou choisir un nom de groupe de journaux existant dans le menu déroulant. Cochez la case pour enregistrer les énoncés de manière sélective si vous enregistrez des fichiers texte de manière sélective.

   > **Note**
   >
   > Activez les journaux texte et/ou audio en cochant la case Enregistrer les énoncés de manière sélective dans les paramètres des journaux de conversation (texte et/ou audio) dans les paramètres d'heure BotAliasde construction. Vous devez configurer le groupe de CloudWatch journaux et le compartiment Amazon S3 pour sélectionner cette option.

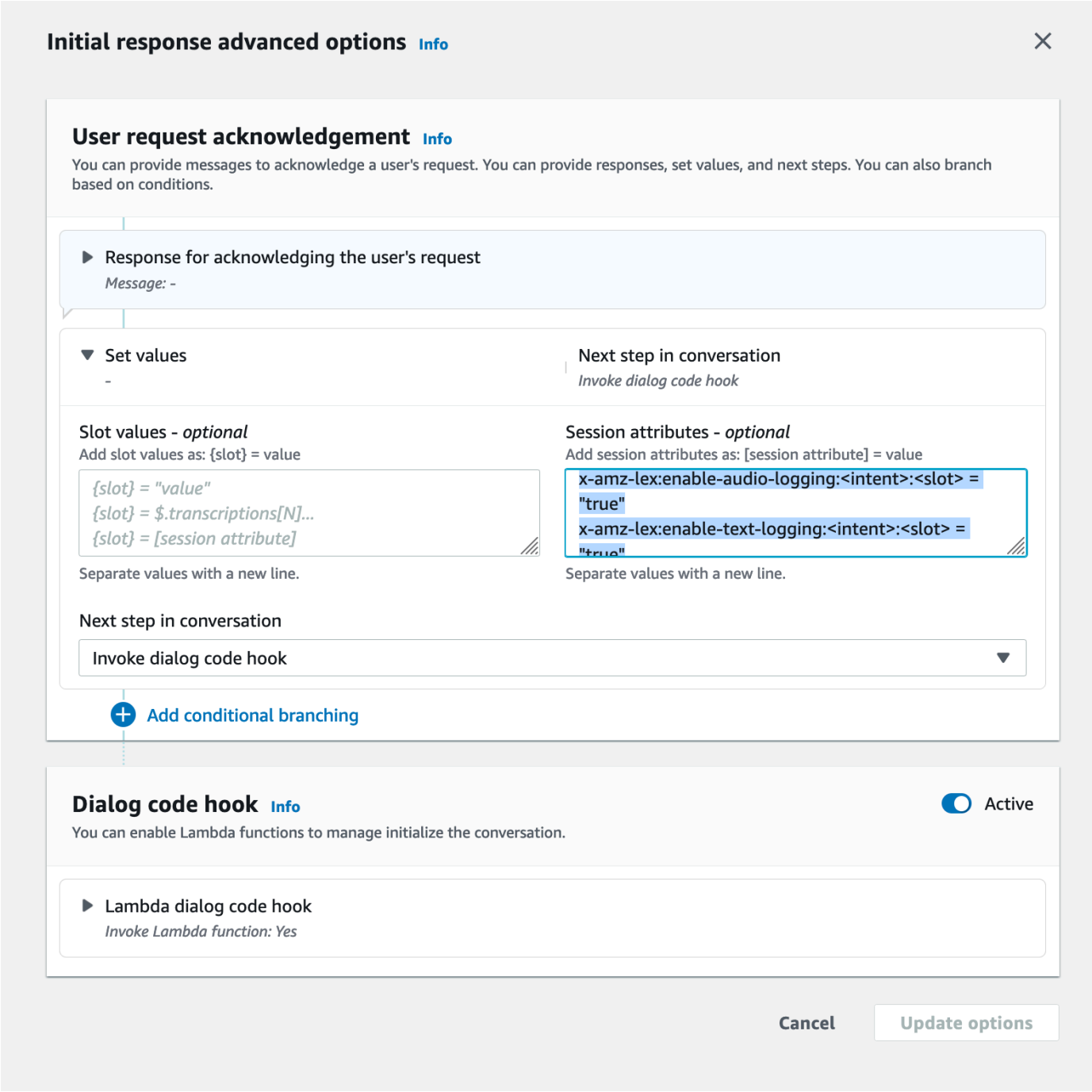
6. Dans la section Journaux audio, choisissez si les journaux audio sont activés ou désactivés en sélectionnant le bouton radio. Si vous choisissez Activé pour les journaux audio, vous devez spécifier l'emplacement du compartiment Amazon S3 et (facultatif) la clé KMS pour chiffrer vos données audio. Cochez la case pour enregistrer les énoncés de manière sélective si vous enregistrez des fichiers audio de manière sélective.



7. Sélectionnez Enregistrer dans le coin inférieur droit du panneau pour enregistrer vos paramètres de capture sélectifs du journal des conversations.

Activez la capture sélective du journal des conversations dans la console Lex :

1. Accédez à Intentions et sélectionnez le nom de l'intention, la réponse initiale, les paramètres avancés, les valeurs définies et les attributs de session.
2. Définissez les attributs suivants en fonction des intentions et des créneaux pour lesquels vous souhaitez activer la capture sélective des journaux de conversation :
   - `x-amz-lex:enable-audio-logging:intent:slot = "true"`
   - `x-amz-lex:enable-text-logging:intent:slot = "true"`



Note

`x-amz-lex:enable-audio-logging:`*`intent`*`:`*`slot`* `= "true"`Réglé pour capturer les énoncés qui ne contiennent qu'un créneau spécifique de la conversation. L'action d'enregistrement d'un énoncé dépend de l'évaluation de l'*`intention`* : *`emplacement`*

dans l'énoncé, par rapport aux expressions des attributs de session, et valeur de l'indicateur correspondant. Pour enregistrer un énoncé, au moins une expression de l'attribut de session doit l'autoriser, l'indicateur d'activation de la journalisation étant défini sur. `true` La valeur de l'*`intention`* et du *`créneau`* peut `"*"` également l'être. Si la valeur d'emplacement et/ou d'intention est égale à cette valeur`"*"`, cela signifie que tout emplacement et/ou valeur d'intention de `"*"` correspondra à cette valeur. De même`x-amz-lex:enable-audio-logging`, un nouvel attribut de session appelé `x-amz-lex:enable-text-logging` sera utilisé pour contrôler les journaux de texte.

3. Sélectionnez Options de mise à jour et créez le bot pour inclure les paramètres mis à jour.

> **Note**
>
> Votre rôle IAM doit disposer d'une autorisation d'accès pour vous permettre d'écrire des données dans le compartiment Amazon S3 et d'utiliser une clé KMS pour chiffrer les données. Lex mettra à jour votre rôle IAM avec les autorisations Lex pour accéder au groupe de CloudWatch journaux Logs et au compartiment Amazon S3 sélectionné.

Instructions relatives à l'utilisation de la capture sélective des journaux de conversation :

Vous ne pouvez activer la capture sélective des journaux de conversation pour les journaux texte et/ou audio que lorsque vous avez activé les journaux texte et/ou audio dans les paramètres du journal des conversations. En activant la capture sélective des journaux de conversation pour les journaux texte et/ou audio, vous désactivez la journalisation pour toutes les fins et tous les créneaux de la conversation. Pour générer des journaux de conversation textuels et/ou audio pour des objectifs et des emplacements particuliers, vous devez définir les attributs de session de capture de journaux de conversation sélectifs en texte et/ou audio pour ces intentions et ces emplacements sur « true ».

- Si la capture sélective du journal des conversations est activée et qu'aucun attribut de session portant le préfixe « x-amz-lex : » enable-audio-logging n'est présent, la journalisation sera désactivée par défaut pour tous les énoncés. Ce scénario est également vrai en ce qui concerne x-amz-lex :enable-text-logging.
- Les journaux d'énoncés seront stockés exclusivement pour les segments de texte et/ou de conversation audio si au moins une expression de l'attribut de session le permet.
- Les configurations pour la capture sélective du texte et/ou de l'audio dans le journal des conversations, telles que définies dans les attributs de session, ne seront efficaces que lorsque

la capture sélective du journal des conversations pour le texte et/ou le son est activée dans les paramètres du journal des conversations dans l'alias du bot ; sinon, les attributs de session ne seront pas pris en compte.

- Lorsque la capture sélective du journal des conversations est activée, toutes les valeurs de créneau dans SessionState les interprétations et les transcriptions pour lesquelles la journalisation n'est pas activée à l'aide des attributs de session seront masquées dans le journal de texte généré.
- La décision de produire des journaux audio et/ou textuels est évaluée en faisant correspondre le créneau obtenu par le bot aux attributs de session de capture sélective des journaux de conversation, à l'exception du tour d'obtention d'intention où l'utilisateur peut fournir des valeurs d'intervalle ainsi que l'obtention d'intentions. Lors d'un tour d'obtention d'intention, les créneaux remplis au tour actuel sont comparés aux attributs de session de capture sélective du journal des conversations.
- Les créneaux considérés comme remplis sont dérivés de l'état de la session à la fin du tour. Par conséquent, toute modification apportée par le Dialog Codehook Lambda aux emplacements dans l'état de session influencera le comportement de la capture sélective des journaux de conversation.
- Lors d'un tour d'obtention d'intention, si plusieurs valeurs de créneaux sont fournies par l'utilisateur, le journal texte et/ou audio ne sera généré que si les attributs de session texte/audio autorisent la journalisation de tous les créneaux remplis lors de ce tour.
- L'approche opérationnelle recommandée consiste à définir l'attribut de session de capture sélective du journal des conversations au début de la session et à ne pas le modifier pendant la session.
- Si des emplacements contiennent des données sensibles, vous devez toujours activer l'obfuscation des emplacements.

Exemple de capture sélective du journal des conversations

Voici un exemple d'utilisation professionnelle pour la capture sélective des journaux de conversation.

Cas d'utilisation :

Une société de technologie financière utilise un bot Amazon Lex V2 pour soutenir son système IVR, qui permet aux utilisateurs d'effectuer des paiements de factures. Afin de répondre aux exigences de conformité et d'audit, ils doivent conserver les enregistrements audio du consentement d'autorisation fourni par l'utilisateur. Cependant, il n'est pas possible d'activer les journaux audio généraux car cela les rendrait non conformes, car il n'est pas possible de masquer les emplacements sensibles tels que CardNumber le CVV et d'autres informations contenues dans les journaux audio. Au lieu de cela, ils peuvent activer la capture sélective des journaux de conversation pour les journaux audio et définir

l'attribut de session pour ne produire des journaux audio que pour les énoncés ayant fait l'objet d'un consentement d'autorisation.

BotAlias Réglages :

- Journaux de texte activés : vrai
- Journaux de texte Enregistrement sélectif activé : faux
- Journaux audio activés : vrai
- Journaux audio Enregistrement sélectif activé : vrai

Attributs de session :

```
x-amz-lex:enable-audio-logging:PayBill:AuthorizationConsent = "true"
```

Exemple de conversation :

- Utilisateur (entrée audio) : « Je souhaite payer ma facture avec le numéro de facture 35XU68. »
- Bot : « Quel est le montant dû en dollars ? »
- Utilisateur (entrée audio) : « 235 »
- Bot : « Quel est votre numéro de carte de crédit ? »
- Utilisateur (entrée audio) : « 9239829722200348. »
- Bot : « Vous payez 235 dollars en utilisant le numéro de votre carte de crédit se terminant par 0348. S'il vous plaît, dites « J'autorise à payer 235 dollars ». »
- Utilisateur (entrée audio) : « J'autorise le paiement de 235 dollars. »
- Bot : « Votre facture a été payée. »

Sortie des journaux de conversation :

Dans ce cas, des journaux de texte seront produits pour tous les tours. Cependant, les journaux audio ne seront enregistrés pour le tour en question que lorsque le AuthorizationConsentcréneau correspondant à l'PayBillintention a été obtenu, et aucun journal audio ne sera produit pour aucun autre tour.

Surveillance des indicateurs opérationnels

Amazon CloudWatch et ce AWS CloudTrail sont deux AWS services qui s'intègrent à Amazon Lex V2 pour vous aider à surveiller les interactions des utilisateurs avec votre bot. Utilisez ces services pour enregistrer les actions, envoyer des données en temps quasi réel et configurer des notifications et des actions automatisées lorsque les critères sont remplis.

Rubriques

Mesurer les indicateurs opérationnels avec Amazon CloudWatch

Vous pouvez surveiller Amazon Lex V2 à l'aide d'Amazon Lex V2 CloudWatch, qui collecte les données brutes et les transforme en indicateurs lisibles en temps quasi réel. Ces statistiques sont enregistrées pour une durée de 15 mois ; par conséquent, vous pouvez accéder aux informations historiques et acquérir un meilleur point de vue de la façon dont votre service ou application web s'exécute. Vous pouvez également définir des alarmes qui surveillent certains seuils et envoient des notifications ou prennent des mesures lorsque ces seuils sont atteints. Pour plus d'informations, consultez le guide de CloudWatch l'utilisateur Amazon.

Le service Amazon Lex V2 indique les métriques suivantes dans l'espace de `AWS/Lex` noms.

Métrique	Description
`AssistedSlotResolutionModelAccessDeniedErrorCount`	Le nombre de fois où Amazon Lex V2 s'est vu refuser l'accès à Amazon Bedrock Dimensions valides pour les `StartConversation` opérations `RecognizeUtterance` et : • BotId, BotAliasId, LocaleId, Opération, InputMode, ModelType, Modèle • BotId, BotVersion, LocaleId, Opération, InputMode, ModelType, Modèle Dimensions valides pour `RecognizeText` :

Métrique	Description
	• BotId, BotAliasId, LocaleId, Opération ModelType, Modèle • BotId, BotVersion, LocaleId, Opération ModelType, Modèle Unité : nombre
`AssistedSlotResolutionModelInvocationCount`	Le nombre de fois où Amazon Bedrock a été invoqué. Dimensions valides pour les `StartConversation` opérations `RecognizeUtterance` et : • BotId, BotAliasId, LocaleId, Opération, InputMode, ModelType, Modèle • BotId, BotVersion, LocaleId, Opération, InputMode, ModelType, Modèle Dimensions valides pour `RecognizeText` : • BotId, BotAliasId, LocaleId, Opération ModelType, Modèle • BotId, BotVersion, LocaleId, Opération ModelType, Modèle Unité : nombre

Métrique	Description
`AssistedSlotResolutionModelSystemErrorCount`	Le nombre de fois où un 5xx s'est produit lors d'un appel à Amazon Bedrock. Dimensions valides pour les `StartConversation` opérations `RecognizeUtterance` et : • BotId, BotAliasId, LocaleId, Opération, InputMode, ModelType, Modèle • BotId, BotVersion, LocaleId, Opération, InputMode, ModelType, Modèle Dimensions valides pour `RecognizeText` : • BotId, BotAliasId, LocaleId, Opération ModelType, Modèle • BotId, BotVersion, LocaleId, Opération ModelType, Modèle Unité : nombre
`AssistedSlotResolutionModelThrottlingErrorCount`	Le nombre de fois où Amazon Lex a été limité par Amazon Bedrock. Dimensions valides pour les `StartConversation` opérations `RecognizeUtterance` et : • BotId, BotAliasId, LocaleId, Opération, InputMode, ModelType, Modèle • BotId, BotVersion, LocaleId, Opération, InputMode, ModelType, Modèle Dimensions valides pour `RecognizeText` : • BotId, BotAliasId, LocaleId, Opération ModelType, Modèle • BotId, BotVersion, LocaleId, Opération ModelType, Modèle Unité : nombre

Métrique	Description
`AssistedSlotResolutionResolvedSlotCount`	Le nombre de fois où Amazon Bedrock a renvoyé une valeur de créneau. Dimensions valides pour les `StartConversation` opérations `RecognizeUtterance` et : • BotId, BotAliasId, LocaleId, Opération, InputMode, ModelType, Modèle • BotId, BotVersion, LocaleId, Opération, InputMode, ModelType, Modèle Dimensions valides pour `RecognizeText` : • BotId, BotAliasId, LocaleId, Opération ModelType, Modèle • BotId, BotVersion, LocaleId, Opération ModelType, Modèle Unité : nombre
`KendraIndexAccessError`	Le nombre de fois où Amazon Lex V2 n'a pas pu accéder à votre index Amazon Kendra. • Fonctionnement BotId, BotAliasId, LocaleId Unité : nombre
`KendraLatency`	Le temps nécessaire à Amazon Kendra pour répondre à une demande du. `AMAZON.KendraSearchIntent` Dimensions valides : • Fonctionnement BotId, BotVersion, LocaleId • Fonctionnement BotId, BotAliasId, LocaleId Unité : millisecondes

Métrique	Description
KendraSuccess	Le nombre de fois où Amazon Lex V2 n'a pas pu accéder à votre index Amazon Kendra. Dimensions valides : • Fonctionnement BotId, BotVersion, LocaleId • Fonctionnement BotId, BotAliasId, LocaleId Unité : nombre
KendraSystemErrors	Le nombre de fois où Amazon Lex V2 n'a pas pu interroger l'index Amazon Kendra. Dimensions valides : • Fonctionnement BotId, BotAliasId, InputMode, LocaleId Unité : nombre
KendraThrottledEvents	Le nombre de fois où Amazon Kendra a limité les demandes provenant du. AMAZON.KendraSearchIntent Dimensions valides : • Opération BotId, BotAliasId, InputMode. LocaleId Unité : nombre

Métrique	Description
`RuntimeConcurrency`	Nombre de connexions simultanées au cours de la période spécifiée. `RuntimeConcurrency` est signalé sous la forme d'un`StatisticSet`. Dimensions valides pour les `StartConversation` opérations `RecognizeUtterance` or : • Fonctionnement BotId, BotVersion, InputMode, LocaleId • Fonctionnement BotId, BotAliasId, InputMode, LocaleId Dimensions valides pour les autres opérations : • Fonctionnement BotId, BotVersion, LocaleId • Fonctionnement BotId, BotAliasId, LocaleId Unité : nombre
`RuntimeInvalidLambdaResponses`	Le nombre de AWS Lambda réponses non valides au cours de la période spécifiée. Dimensions valides : • Fonctionnement BotId, BotAliasId, InputMode, LocaleId Unité : nombre
`RuntimeLambdaErrors`	Nombre d'erreurs d'exécution Lambda au cours de la période spécifiée. Dimensions valides : • Fonctionnement BotId, BotAliasId, InputMode, LocaleId Unité : nombre

Métrique	Description
`RuntimePollyErrors`	Le nombre de réponses Amazon Polly non valides au cours de la période spécifiée. Dimensions valides : • Fonctionnement BotId, BotAliasId, InputMode, LocaleId Unité : nombre
`RuntimeRequestCount`	Le nombre de demandes d'exécution au cours de la période spécifiée. Dimensions valides pour les `StartConversation` opérations `RecognizeUtterance` et : • Fonctionnement BotId, BotVersion, InputMode, LocaleId • Fonctionnement BotId, BotAliasId, InputMode, LocaleId Dimensions valides pour les autres opérations : • Fonctionnement BotId, BotVersion, LocaleId • Fonctionnement BotId, BotAliasId, LocaleId Unité : nombre
`RuntimeRequestLength`	Durée totale d'une conversation avec un bot Amazon Lex V2. Applicable uniquement à l'StartConversationopération. Dimensions valides : • BotAliasId, BotId, LocaleId, Opération • BotId, BotAliasId, LocaleId, Opération Unité : millisecondes

Métrique	Description
`RuntimeSuccessfulRequestLatency` **Important** Cette métrique l'est `RuntimeSuccessfulRequestLatency` et ne l'est pas`RuntimeSuccessfulRequestLatency` .	Le temps de latence des demandes réussies entre le moment où la demande a été faite et le moment où la réponse a été renvoyée. Dimensions valides pour les `StartConversation` opérations `RecognizeUtterance` et : • Fonctionnement BotId, BotVersion, InputMode, LocaleId • Fonctionnement BotId, BotAliasId, InputMode, LocaleId Dimensions valides pour les autres opérations : • Fonctionnement BotId, BotVersion, LocaleId • Fonctionnement BotId, BotAliasId, LocaleId Unité : millisecondes
`RuntimeSystemErrors`	Nombre d'erreurs système au cours de la période spécifiée. La plage des codes de réponse d'une erreur système est comprise entre 500 et 599. Dimensions valides pour les `StartConversation` opérations `RecognizeUtterance` et : • Fonctionnement BotId, BotVersion, InputMode, LocaleId • Fonctionnement BotId, BotAliasId, InputMode, LocaleId Dimensions valides pour les autres opérations : • Fonctionnement BotId, BotVersion, LocaleId • Fonctionnement BotId, BotAliasId, LocaleId Unité : nombre

Métrique	Description
`RuntimeThrottledEvents`	Le nombre d'événements limités. Amazon Lex V2 limite un événement lorsqu'il reçoit plus de demandes que la limite de transactions par seconde définie pour votre compte. Si cette limite est souvent franchie, vous pouvez demander une augmentation de la limite. Pour demander une augmentation, consultez les limites de service AWS. Dimensions valides pour les `StartConversation` opérations `RecognizeUtterance` et : • Fonctionnement BotId, BotVersion, InputMode, LocaleId • Fonctionnement BotId, BotAliasId, InputMode, LocaleId Dimensions valides pour les autres opérations : • Fonctionnement BotId, BotVersion, LocaleId • Fonctionnement BotId, BotAliasId, LocaleId Unité : nombre

Métrique	Description
`RuntimeUserErrors`	Nombre d'erreurs utilisateur au cours de la période spécifiée. La plage des codes de réponse d'une erreur d'utilisateur est comprise entre 400 et 499. Dimensions valides pour les `StartConversation` opérations `RecognizeUtterance` et : • Fonctionnement BotId, BotVersion, InputMode, LocaleId • Fonctionnement BotId, BotAliasId, InputMode, LocaleId Dimensions valides pour les autres opérations : • Fonctionnement BotId, BotVersion, LocaleId • Fonctionnement BotId, BotAliasId, LocaleId Unité : nombre

Les dimensions suivantes sont prises en charge pour les métriques Amazon Lex V2.

Dimension	Description
`Operation`	Nom de l'opération Amazon Lex V2 —`RecognizeText` ,`RecognizeUtterance` ,`StartConversation` ,`GetSession` ,`PutSession` , `DeleteSession` — qui a généré l'entrée.
`BotId`	Identifiant unique alphanumérique du bot.
`BotAliasId`	Identifiant unique alphanumérique de l'alias du bot.
`BotVersion`	Version numérique du bot.
`InputMode`	Type de saisie du bot : voix, texte ou DTMF.
`LocaleId`	L'identifiant de la localisation du bot, tel que en-US ou fr-CA.

Dimension	Description
`Model`	Indique l'identifiant du modèle de langage large Amazon Bedrock.
`ModelType`	Indique le type de grand modèle de langage invoqué depuis Amazon Bedrock.

Afficher les événements avec AWS CloudTrail

Amazon Lex V2 est intégré à AWS CloudTrail un service qui fournit un enregistrement des actions entreprises par un utilisateur, un rôle ou un AWS service dans Amazon Lex V2. CloudTrail capture les appels d'API pour Amazon Lex V2 sous forme d'événements. Les appels capturés incluent des appels provenant de la console Amazon Lex V2 et des appels de code vers les opérations de l'API Amazon Lex V2. Si vous créez un suivi, vous pouvez activer la diffusion continue d' CloudTrail événements vers un compartiment Amazon S3, y compris des événements pour Amazon Lex V2. Si vous ne configurez pas de suivi, vous pouvez toujours consulter les événements les plus récents dans la CloudTrail console dans Historique des événements. À l'aide des informations collectées par CloudTrail, vous pouvez déterminer la demande envoyée à Amazon Lex V2, l'adresse IP à partir de laquelle la demande a été faite, l'auteur de la demande, la date à laquelle elle a été faite, ainsi que des informations supplémentaires.

Pour en savoir plus CloudTrail, consultez le guide de AWS CloudTrail l'utilisateur.

Informations sur Amazon Lex V2 dans CloudTrail

CloudTrail est activé sur votre AWS compte lorsque vous le créez. Lorsqu'une activité se produit dans Amazon Lex V2, cette activité est enregistrée dans un CloudTrail événement avec d'autres événements de AWS service dans l'historique des événements. Vous pouvez consulter, rechercher et télécharger les événements récents dans votre AWS compte. Pour plus d'informations, consultez la section Affichage des événements à l'aide de l'historique des CloudTrail événements.

Pour un enregistrement continu des événements de votre AWS compte, y compris des événements relatifs à Amazon Lex V2, créez un historique. Un suivi permet CloudTrail de fournir des fichiers journaux à un compartiment Amazon S3. Par défaut, lorsque vous créez un journal d'activité dans la console, il s’applique à toutes les régions AWS. Le journal enregistre les événements de toutes les régions de la AWS partition et transmet les fichiers journaux au compartiment Amazon S3 que vous spécifiez. En outre, vous pouvez configurer d'autres AWS services pour analyser plus en détail les

données d'événements collectées dans les CloudTrail journaux et agir en conséquence. Pour plus d'informations, consultez les ressources suivantes :

- Présentation de la création d'un journal de suivi
- CloudTrail services et intégrations pris en charge
- Configuration des notifications Amazon SNS pour CloudTrail
- Réception de fichiers CloudTrail journaux de plusieurs régions et réception de fichiers CloudTrail journaux de plusieurs comptes

Amazon Lex V2 prend en charge la journalisation de toutes les actions répertoriées dans l'API Model Building V2.

Chaque événement ou entrée de journal contient des informations sur la personne ayant initié la demande. Les informations relatives à l'identité permettent de déterminer :

- Si la demande a été faite avec les informations d'identification de l'utilisateur root ou AWS Identity and Access Management IAM.
- Si la demande a été effectuée avec les informations d'identification de sécurité temporaires d'un rôle ou d'un utilisateur fédéré.
- Si la demande a été faite par un autre AWS service.

Pour plus d'informations, consultez l'élément CloudTrail UserIdentity.

Comprendre les entrées du fichier journal Amazon Lex V2

Un suivi est une configuration qui permet de transmettre des événements sous forme de fichiers journaux à un compartiment Amazon S3 que vous spécifiez. CloudTrail les fichiers journaux contiennent une ou plusieurs entrées de journal. Un événement représente une demande unique provenant de n'importe quelle source et inclut des informations sur l'action demandée, la date et l'heure de l'action, les paramètres de la demande, etc. CloudTrail les fichiers journaux ne constituent pas une trace ordonnée des appels d'API publics, ils n'apparaissent donc pas dans un ordre spécifique.

L'exemple suivant montre une entrée de CloudTrail journal illustrant l'CreateBotAliasaction.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
```

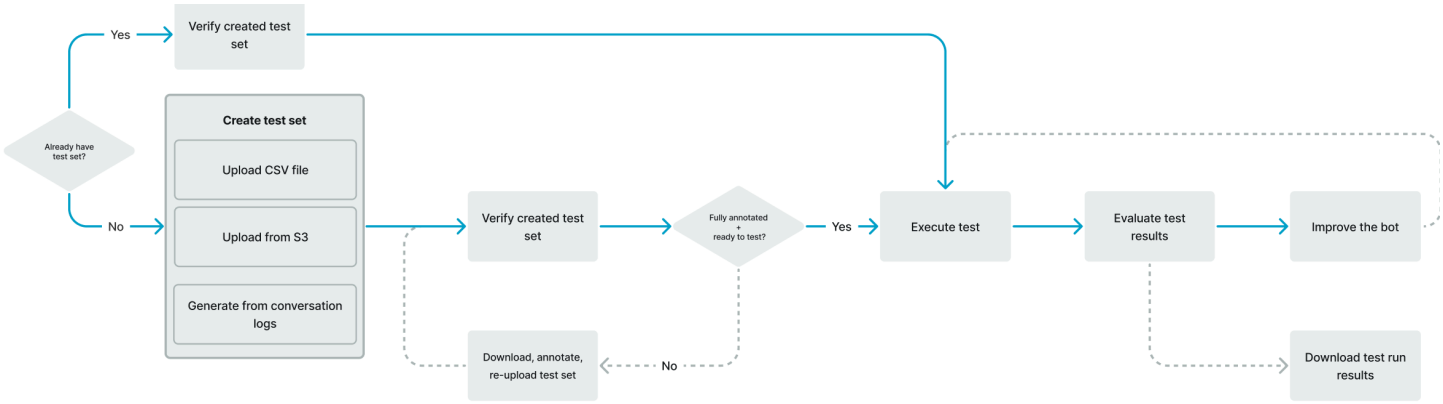
```
        "type": "AssumedRole",
        "principalId": "ID of caller:temporary credentials",
        "arn": "arn:aws:sts::111122223333:assumed-role/role name/role ARN",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "ID of caller",
                "arn": "arn:aws:iam::111122223333:role/role name",
                "accountId": "111122223333",
                "userName": "role name"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "creation date"
            }
        }
    },
    "eventTime": "event timestamp",
    "eventSource": "lex.amazonaws.com",
    "eventName": "CreateBotAlias",
    "awsRegion": "Region",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "user agent",
    "requestParameters": {
        "botAliasLocaleSettingsMap": {
            "en_US": {
                "enabled": true
            }
        },
        "botId": "bot ID",
        "botAliasName": "bot aliase name",
        "botVersion": "1"
    },
    "responseElements": {
        "botAliasLocaleSettingsMap": {
            "en_US": {
                "enabled": true
            }
        },
        "botAliasId": "bot alias ID",
        "botAliasName": "bot alias name",
```

```
        "botId": "bot ID",
        "botVersion": "1",
        "creationDateTime": creation timestamp
    },
    "requestID": "unique request ID",
    "eventID": "unique event ID",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

Évaluation des performances des robots avec le Test Workbench

Pour améliorer les performances des robots, vous pouvez évaluer les performances de vos robots à grande échelle. Les résultats de l'évaluation de votre test sont affichés sous forme de tableaux et de graphiques simples.

Vous pouvez utiliser le banc de test pour créer des ensembles de tests de référence utilisant des données de transcription existantes. Vous pouvez tester les robots pour évaluer les performances avant le déploiement et consulter la ventilation des résultats des tests à grande échelle.

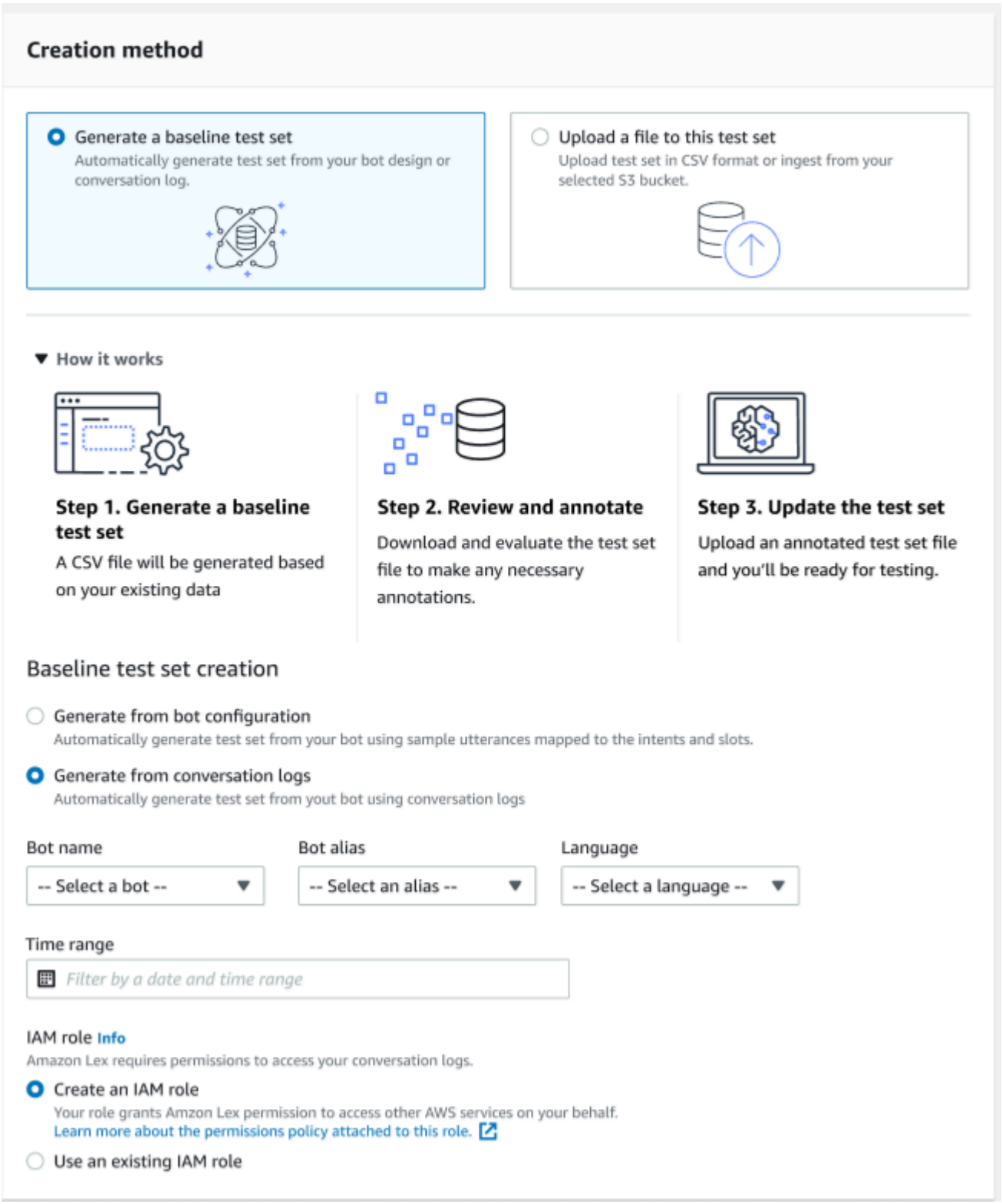


Les utilisateurs peuvent utiliser le banc de test pour établir les performances de référence des robots. Cela couvre l'intention et les performances des créneaux pour les énoncés qui prennent la forme d'entrées uniques ou de conversations. Une fois qu'un ensemble de tests est chargé avec succès, vous pouvez l'exécuter sur vos robots de pré-production ou de production existants. Le banc de test vous aide à identifier les opportunités d'amélioration du remplissage des emplacements et de la classification des intentions.

Rubriques

Génération d'un ensemble de tests

Vous pouvez créer un ensemble de tests pour évaluer les performances de votre bot. Générez un ensemble de tests en téléchargeant un ensemble de tests au format de fichier CSV ou en générant un ensemble de tests à partir des journaux de conversation. Le kit de test peut contenir une entrée audio ou textuelle.



Si un ensemble de tests crée des erreurs de validation, supprimez-le et remplacez-le par une autre liste de données d'ensemble de tests, ou modifiez les données du fichier CSV à l'aide d'un programme d'édition de feuille de calcul.

Pour créer un ensemble de tests, procédez comme suit :

1. Connectez-vous à la console Amazon Lex AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/lex/.
2. Choisissez Test workbench dans le panneau de gauche.
3. Sélectionnez Ensembles de tests dans les options sous Test workbench.
4. Cliquez sur le bouton Créer un ensemble de tests sur la console.
5. Dans les détails, entrez le nom du set de test et une description facultative.
6. Sélectionnez Générer un ensemble de tests de référence.
7. Sélectionnez Générer à partir des journaux de conversation.
8. Sélectionnez le nom du bot, l'alias du bot et la langue dans les menus déroulants.
9. Si vous générez un test de référence à partir d'un journal de conversation, choisissez Plage de temps et rôle IAM, si nécessaire. Vous pouvez créer un rôle avec les autorisations de base d'Amazon Lex V2 ou utiliser un rôle existant.
10. Choisissez une modalité audio ou texte pour le set de test que vous créez. REMARQUE : Le Test Workbench peut importer des fichiers texte jusqu'à 50 000 et jusqu'à 5 heures d'audio.
11. Sélectionnez un emplacement Amazon S3 pour stocker les résultats de vos tests et ajoutez une clé KMS facultative pour chiffrer les transcriptions de sortie.
12. Sélectionnez Créer.

Pour télécharger un ensemble de tests existant dans un format de fichier CSV ou pour le mettre à jour :

1. Choisissez Test workbench dans le panneau de gauche.
2. Sélectionnez Ensembles de tests dans les options sous Test workbench.
3. Sélectionnez Charger un fichier vers cet ensemble de test sur la console.
4. Choisissez Upload depuis le bucket Amazon S3 ou Upload depuis votre ordinateur. REMARQUE : Vous pouvez télécharger un fichier CSV créé à partir d'un modèle. Cliquez sur Modèle CSV pour télécharger un fichier zip contenant les modèles.
5. Choisissez Créer un rôle avec des autorisations Amazon Lex de base ou Utiliser un rôle existant pour l'ARN du rôle.
6. Choisissez une modalité audio ou texte pour le set de test que vous créez. REMARQUE : Le Test Workbench peut importer des fichiers texte jusqu'à 50 000 et jusqu'à 5 heures d'audio.

7. Sélectionnez un emplacement Amazon S3 pour stocker les résultats de vos tests et ajoutez une clé KMS facultative pour chiffrer les transcriptions de sortie.
8. Sélectionnez Créer.

Si l'opération est réussie, le message de confirmation indique que le kit de test est prêt à être testé, et le statut indique Prêt pour le test.

Conseils pour créer un ensemble de test réussi

- Vous pouvez créer un rôle IAM pour le Test Workbench dans la console ou configurer votre rôle IAM. step-by-step Pour plus d'informations, voir Créer un rôle IAM pour le Test Workbench.
- Avant d'exécuter un test, validez l'ensemble de tests et la définition du bot pour détecter toute incohérence à l'aide du bouton Valider l'écart. Si les conventions d'intention et de dénomination des emplacements utilisées dans le set de test sont cohérentes avec celles du bot, procédez à l'exécution du test. Si des anomalies sont identifiées, révisez le jeu de tests, mettez-le à jour et choisissez Valider l'écart. Répétez cette séquence jusqu'à ce qu'aucune incohérence ne soit constatée, puis exécutez le test.
- Le banc de test peut effectuer des tests avec différents formats de valeur d'emplacement dans la colonne Emplacement de sortie attendu. Pour n'importe quel emplacement intégré, vous pouvez choisir la valeur fournie dans l'entrée utilisateur (par exemple, Date = demain) ou fournir sa valeur absolue résolue (par exemple, Date = ↓ 21). Pour plus d'informations sur les emplacements intégrés et leurs valeurs absolues, consultez la section Emplacements intégrés.
- Pour des raisons de cohérence et de lisibilité dans les colonnes de l'emplacement de sortie attendu, suivez la convention SlotValue « SlotName = » (par exemple, AppointmentType = nettoyage) avec un espace avant et après le signe égal.
- Si le bot inclut des emplacements composites, dans Expected Output Slot, définissez les sous-emplacements correspondant au nom de l'emplacement, séparés par un point (par exemple, « Car.Color »). Aucune autre syntaxe ou ponctuation ne fonctionnera.
- Si le bot inclut des emplacements à valeurs multiples, dans Expected Output Slot, fournissez plusieurs valeurs d'emplacements, séparées par une virgule (« FlowerType = roses, lys »). Aucune autre syntaxe ou ponctuation ne fonctionnera.
- Assurez-vous que l'ensemble de tests est créé à partir de journaux de conversation valides.
- SLOT:La valeur de l'emplacement sera dans la même colonne après les colonnes d'intention au format CSV.

- L'entrée DTMF provenant du tour d'un utilisateur est interprétée comme une transcription attendue et ne répertorie aucun emplacement Amazon S3.

Création d'un scénario de test dans un ensemble de tests

Les résultats du Test Workbench dépendent de la définition du bot et de l'ensemble de tests correspondant. Vous pouvez générer un ensemble de tests à partir des informations issues de la définition du bot afin d'identifier les domaines à améliorer. Créez un jeu de données de test avec des exemples que vous pensez (ou savez) que le bot aura du mal à interpréter correctement, compte tenu de la conception actuelle du bot et de votre connaissance des conversations avec les clients.

Passez régulièrement en revue vos intentions en vous basant sur les leçons apprises par votre robot de production. Continuez à ajouter et à ajuster les exemples d'énoncés et les valeurs des créneaux du bot. Envisagez d'améliorer la résolution des emplacements en utilisant les options disponibles, telles que les indicateurs d'exécution. La conception et le développement de votre bot sont un processus itératif qui est un cycle continu.

Voici d'autres conseils pour optimiser votre ensemble de tests :

- Sélectionnez les cas d'utilisation les plus courants avec des intentions et des emplacements fréquemment utilisés dans le kit de test.
- Découvrez les différentes manières dont un client pourrait se référer à vos intentions et à vos créneaux. Cela peut inclure des entrées utilisateur sous forme d'énoncés, de questions et de commandes dont la longueur varie de minimale à étendue.
- Incluez les entrées utilisateur avec un nombre varié d'emplacements.
- Incluez des synonymes ou des abréviations couramment utilisés des valeurs de créneaux personnalisées prises en charge par votre bot (par exemple, « canal racine », « canal » ou « RC »).
- Incluez des variations des valeurs de créneau intégrées (par exemple, « demain », « dès que possible » ou « le jour suivant »).
- Examinez la robustesse du bot par rapport à la modalité vocale en collectant les entrées utilisateur susceptibles d'être mal interprétées (par exemple, « encre », « cheville » ou « ancre »).

Création d'un ensemble de tests à partir d'un fichier CSV

Vous pouvez créer un ensemble de tests à partir du modèle de fichier CSV fourni dans la console Amazon Lex V2 en saisissant les valeurs directement à l'aide d'un éditeur de feuille de calcul CSV. L'ensemble de tests est un fichier de valeurs séparées par des virgules (CSV) composé d'énoncés

émis par un seul utilisateur et de conversations à plusieurs tours enregistrés dans les colonnes suivantes :

- Numéro de ligne : cette colonne est un compteur incrémentiel qui permet de suivre le nombre total de lignes remplies à tester.
- N° de conversation : cette colonne indique le nombre de tours d'une conversation. Pour les entrées uniques, cette colonne peut être laissée vide, remplie avec « - » ou « N/A ». Pour les conversations, le même numéro de conversation sera attribué à chaque tour d'une conversation.
- Source : cette colonne est définie sur « Utilisateur » ou « Agent ». Pour les entrées uniques, il sera toujours réglé sur « Utilisateur ».
- Saisie : cette colonne inclut l'énoncé de l'utilisateur ou les instructions du bot.
- Intention de sortie attendue : cette colonne indique l'intention remplie dans l'entrée.
- Intention : emplacement de sortie attendu 1 : cette colonne capture le premier emplacement indiqué dans l'entrée utilisateur. L'ensemble de tests doit inclure une colonne intitulée Expected Output Slot X pour chaque slot de l'entrée utilisateur.

Exemple d'un ensemble de test avec entrées uniques :

N° de ligne	Conversation no	Source	Entrée	Objectif de sortie attendu	Emplacement de sortie attendu 1	Emplacement de sortie attendu 2
1		Utilisateur	prenez rendez-vous pour le nettoyage demain	MakeAppointment	AppointmentType = nettoyage	Date = demain
2	N/A	Utilisateur	prenez rendez-vous pour le nettoyage le 15 avril	MakeAppointment	AppointmentType = nettoyage	Date = 15/04/23
3	N/A	Utilisateur	prendre rendez-	MakeAppointment	Date = 1er décembre	

N° de ligne	Conversation no	Source	Entrée	Objectif de sortie attendu	Emplacement de sortie attendu 1	Emplacement de sortie attendu 2
			vous pour le 1er décembre			
4	N/A	Utilisateur	prendre un rendez-vous de nettoyage	MakeAppointment	AppointmentType = nettoyage	
1		Utilisateur	Pouvez-vous m'aider à prendre rendez-vous ?	MakeAppointment		

Exemple d'un ensemble de tests avec conversations

N° de ligne	Conversation no	Source	Entrée	Objectif de sortie attendu	Emplacement de sortie attendu 1	Emplacement de sortie attendu 2	Emplacement de sortie attendu 3
1	1	Utilisateur	prendre rendez-vous	MakeAppointment			
2	1	Agent	Quel type de rendez-vous souhaitez-vous prendre ?	MakeAppointment			

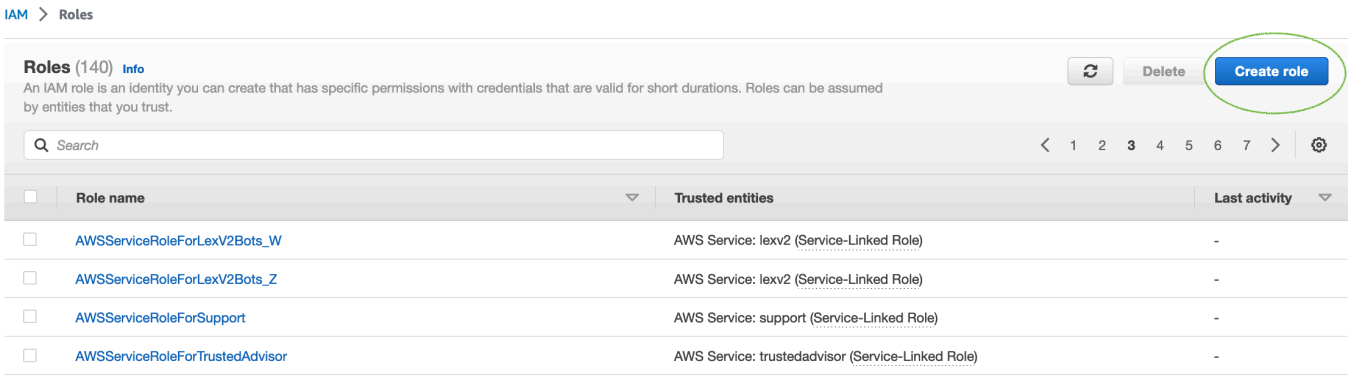
N° de ligne	Conversation no	Source	Entrée	Objectif de sortie attendu	Emplacement de sortie attendu 1	Emplacement de sortie attendu 2	Emplacement de sortie attendu 3
3	1	Utilisateur	nettoyant	MakeAppointment	AppointmentType = nettoyage		
4	1	Agent	Quand dois-je fixer votre rendez-vous ?	MakeAppointment			
5	1	Utilisateur	tomorrow	MakeAppointment		Date = demain	
6	2	Utilisateur	prenez rendez-vous pour un canal radiculaire dès aujourd'hui	MakeAppointment	AppointmentType = canal radiculaire	Date = aujourd'hui	
7	2	Agent	À quelle heure dois-je prendre rendez-vous ?	MakeAppointment			

N° de ligne	Conversation no	Source	Entrée	Objectif de sortie attendu	Emplacement de sortie attendu 1	Emplacement de sortie attendu 2	Emplacement de sortie attendu 3
8	2	Utilisateur	onze heures du matin	MakeAppointment			Heure = onze heures

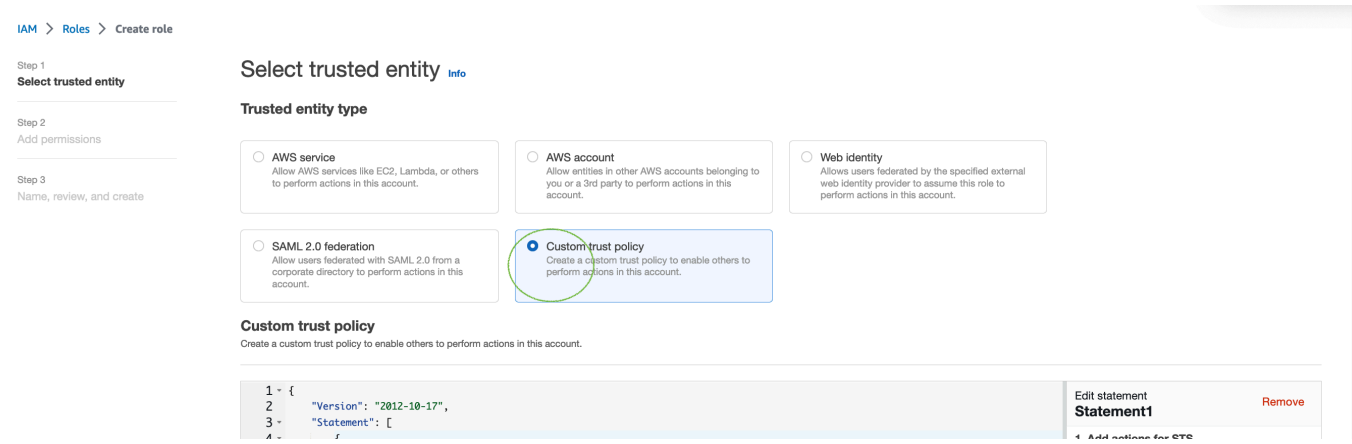
Création d'un rôle IAM pour le Test Workbench

Pour créer un rôle IAM pour le Test Workbench

1. Suivez les étapes de la section Créer un utilisateur IAM pour créer un utilisateur IAM qui peut être utilisé pour accéder à la console test-workbench.
2. Sélectionnez le bouton Créer un rôle.



3. Sélectionnez l'option Politique de confiance personnalisée.



4. Entrez la politique de confiance ci-dessous et cliquez sur Suivant.

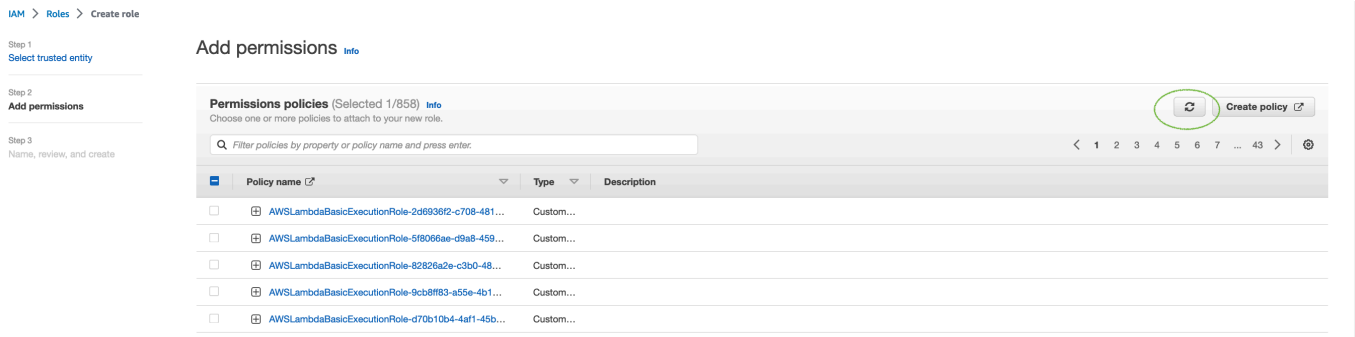
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "sid4",
      "Effect": "Allow",
      "Principal": {
        "Service": "lexv2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

5. Sélectionnez le bouton Créer une politique.
6. Un nouvel onglet s'ouvrira dans votre navigateur où vous pourrez saisir la politique ci-dessous et cliquer sur le bouton Suivant : Tags.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:FilterLogEvents"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "lex:*"
            ],
            "Resource": "*"
```

```
            }
        ]
}
```

7. Entrez un nom de politique, par exemple « LexTestWorkbenchPolicy », puis cliquez sur Créer une politique.
8. Retournez à l'onglet précédent de votre navigateur et actualisez la liste des politiques en cliquant sur le bouton Actualiser comme indiqué ci-dessous.



9. Recherchez dans la liste des politiques en saisissant le nom de la politique que vous avez utilisé à la 6e étape et en choisissant la politique.
10. Cliquez sur le bouton Suivant.
11. Entrez le nom du rôle, puis cliquez sur le bouton Créer un rôle.
12. Choisissez votre nouveau rôle IAM lorsque vous y êtes invité dans la console Amazon Lex V2 pour Test Workbench.

Gérer les ensembles de tests

Vous pouvez télécharger, mettre à jour et supprimer des ensembles de tests depuis la fenêtre des ensembles de tests. Vous pouvez également utiliser la liste des ensembles de tests disponibles pour modifier ou annoter manuellement le fichier de votre ensemble de tests. Ensuite, téléchargez-le à nouveau pour réessayer de le valider, en raison d'erreurs ou d'autres problèmes de saisie.

Pour télécharger le fichier du set de test à partir de l'enregistrement du set de test :

1. Sélectionnez le nom du set de test dans la liste des ensembles de test.
2. Dans la fenêtre d'enregistrement du set de test, sélectionnez le bouton Télécharger sur le côté droit de l'écran dans la section Entrées de test.
3. si des informations relatives à l'erreur de validation apparaissent en haut de la fenêtre concernant le set de test, cliquez sur le bouton Télécharger. Le fichier sera enregistré dans votre

dossier Téléchargements. Vous pouvez corriger les erreurs de validation dans le jeu de tests à partir des messages d'erreur contenus dans le fichier CSV du jeu de tests. Trouvez l'erreur identifiée lors de l'étape de validation, corrigez la ligne ou supprimez-la, puis téléchargez le fichier pour réessayer l'étape de validation.

4. si vous téléchargez le kit de test avec succès, une bannière verte apparaîtra.

Pour télécharger un ensemble de tests à partir de la liste des ensembles de test :

1. Dans la liste des ensembles de test, sélectionnez le bouton radio situé à côté de l'élément du kit de test que vous souhaitez télécharger.
2. Dans le menu Action en haut à droite, choisissez Télécharger.
3. Un message en forme de bannière verte indiquera si vous avez correctement téléchargé le kit de test. Le fichier sera enregistré dans votre dossier Téléchargements.

Afficher les erreurs de validation des tests

Vous pouvez corriger les ensembles de tests qui signalent des erreurs de validation. Ces erreurs de validation sont générées lorsqu'un ensemble de test n'est pas prêt à être testé. Le banc de test peut vous indiquer quelles colonnes requises du fichier CSV d'entrée du jeu de tests ne contenaient pas de valeur au format attendu.

Pour afficher les erreurs de validation des tests :

1. Dans la liste des ensembles de tests, sélectionnez le nom du set de test qui signale un état d'erreur de validation que vous souhaitez consulter. Les noms des ensembles de test sont des liens actifs qui vous permettent d'accéder aux informations concernant le set de test.
2. L'enregistrement du set de test affiche le détail des erreurs de validation en haut de l'écran. Choisissez Afficher les détails pour voir le rapport sur les erreurs de validation.
3. Dans la fenêtre du rapport d'erreur, passez en revue le numéro de ligne et le type d'erreur pour voir où l'erreur se produit. Pour obtenir une longue liste d'erreurs, vous pouvez choisir de télécharger le rapport d'erreurs.
4. Comparez les erreurs répertoriées dans le fichier CSV d'entrée de votre ensemble de tests à votre fichier de test d'origine pour corriger les éventuels problèmes et chargez à nouveau le set de test.

Le tableau suivant répertorie les messages d'erreur de validation CSV en entrée avec les scénarios.

Scénario	Message d’erreur	Remarques
La taille du fichier de l'ensemble de tests dépasse	La taille du fichier Test Set est supérieure à 200 Mo. Fournissez un fichier plus petit et réessayez votre demande.	
L'ensemble de tests dépasse le nombre maximum d'enregistrements	Le fichier d'entrée contenait plus d'enregistrements que le nombre maximum autorisé de 200 000.	
Télécharger un set de test vide	Le set de test importé est vide. Fournissez un ensemble de test non vide et réessayez votre demande.	
Nom d'en-tête de colonne vide	Ligne d'en-têtes de colonne : nom de colonne vide trouvé dans la colonne numéro 5.	
Nom d'en-tête de colonne non reconnu	Ligne d'en-têtes de colonne : impossible de reconnaître le nom de colonne « factice » dans la colonne numéro 2.	
Nom d'en-tête de colonne dupliqué	Ligne d'en-têtes de colonne : plusieurs colonnes « Lien audio S3 » et « Lien audio S3 » ont été trouvées qui sont identiques ou équivalentes. Supprimez ou renommez l'une de ces colonnes.	
Le nom de colonne à valeurs multiples a dépassé la limite	Ligne d'en-têtes de colonne : le nombre de colonnes pour « Emplacement de sortie attendu » a dépassé	Le nombre maximum de colonnes prises en charge pour les colonnes à valeurs multiples est de 6.

Scénario	Message d'erreur	Remarques
	le nombre maximum pris en charge : 6. Supprimez certaines colonnes pour « Emplacement de sortie attendu » et réessayez.	
L'en-tête de colonne lié au texte ou à l'audio n'est pas présent	Impossible de trouver des colonnes pour les conversations textuelles ou audio. Pour les conversations textuelles, utilisez les colonnes {'Text input'}. Pour les conversations audio, utilisez les colonnes {« Lien audio S3 », « Transcription attendue »}.	Colonnes audio obligatoires : {« Lien audio S3 », « Transcription attendue »} Colonnes de texte obligatoires : {'Entrée de texte'}
Les en-têtes de colonne liés au texte et à l'audio existent	Colonnes trouvées pour les conversations textuelles et audio. Vous pouvez soit utiliser les colonnes {'Entrée de texte'} pour les conversations textuelles, soit les colonnes {'Lien audio 3', 'Transcription attendue'} pour les conversations audio.	Colonnes audio obligatoires : {« Lien audio S3 », « Transcription attendue »} Colonnes de texte obligatoires : {'Entrée de texte'}
Colonne obligatoire manquante	Impossible de trouver les colonnes obligatoires ["Intention de sortie attendue"].	Colonnes obligatoires : {"Numéro de ligne », « Source », « Intention de sortie attendue"}

Scénario	Message d’erreur	Remarques
Une donnée a été trouvée dans une colonne sans en-tête	Des données ont été trouvées dans la colonne numéro 8 pour la ligne numéro 6, mais la colonne correspondante ne comportait pas d'en-tête de colonne.	
Données introuvables pour les colonnes obligatoires	Row=12 : aucune valeur n'a été trouvée pour les colonnes obligatoires : {"Source », « Intention de sortie attendue"}	
Identifiant de conversation dupliqué détecté	le numéro de conversation « 19 » a été vu pour une conversation précédente à la ligne numéro 39. » Assurez-vous que le même numéro de conversation n'a pas été fourni pour deux conversat ions. Pour ce faire, assurez-v ous que toutes les lignes d'un numéro de conversation sont regroupées.	
Identifiant de conversation non valide fourni	La valeur « test_conv ersation » non valide a été trouvée dans la colonne « Conversation # ». La valeur de cette colonne doit être numérique ou N/A (c'est-à-dire non applicable) pour une ligne utilisateur.	

Scénario	Message d'erreur	Remarques
Valeur non numérique fournie pour le numéro de ligne	La valeur non numérique « test_line » a été trouvée dans la colonne « Numéro de ligne ». Sa valeur doit être numérique.	
Identifiant de conversation introuvable dans la ligne des agents	Aucune valeur n'a été trouvée pour la colonne « Conversation # ». Il doit être fourni pour une ligne d'agent.	
Identifiant de conversation non numérique trouvé dans la ligne de l'agent	La valeur non numérique « test_conversation » a été trouvée dans la colonne « Conversation # ». Sa valeur doit être numérique pour une ligne d'agent.	
Emplacement S3 non valide	La valeur « bucket/folder » non valide a été fournie. Le format valide est S3 ://<bucketName>/<keyName>.	
Nom de compartiment S3 non valide	Le nom de compartiment s3 « test_bucket » non valide a été fourni. Vérifiez le nom du compartiment.	
L'emplacement audio S3 est un dossier	L'emplacement audio fourni « S3 : //bucket/folder » n'est pas valide. Il pointe vers un dossier S3.	

Scénario	Message d'erreur	Remarques
Nom d'intention non valide	Des caractères non valides étaient présents dans l'intenti on 'intent @name '. Vérifiez le nom de l'intention.	Vérification Regex : ^ ([0-9a-Za-Z] [_-] ?) +$
Nom de slot non valide	Des caractères non valides étaient présents dans l'emplacement « Slot @Name ». Vérifiez le nom du slot.	Régex : ^ ([0-9a-Za-Z] [_-] ?) + $Il ne doit pas commencer ou se terminer par un point (.)
Valeur d'emplacement fournie pour l'emplacement parent	Des valeurs d'emplacement ont été fournies pour le sous-emplacement « Address.City » ainsi que pour le slot parent « Address ». Les valeurs ne doivent être fournies que pour le sous-slot.	L'emplacement parent dans CST ne doit pas avoir de valeur d'emplacement
Caractère non valide dans le nom du contexte	Des caractères non valides étaient présents dans le nom de contexte « context @1 ». Vérifiez le nom du contexte.	Régex : ^ ([A-za-Z] _ ?) +$
Style d'orthographe de slot non valide	La valeur « test » non valide a été fournie. Assurez-vous qu'ils sont tous en majuscule s. Les valeurs valides sont ["Default », "SpellByLetter«," SpellByWord «].	Valeurs prises en charge ["Default », "SpellByLetter«," SpellByWord »
Le participant ou la source doit être un agent ou un utilisateur	La valeur « bot » non valide a été fournie. Les valeurs valides sont ["Agent », « User"].	Enums pris en charge : « Agent », « Utilisateur »

Scénario	Message d'erreur	Remarques
Le numéro de ligne ne doit pas être décimal	La valeur « 10.1 » non valide a été fournie. Il doit s'agir d'un nombre valide sans aucune fraction.	
Le numéro de conversation ne doit pas être décimal	La valeur « 10.1 » non valide a été fournie. Il doit s'agir d'un nombre valide sans aucune fraction.	
Le numéro de ligne doit être compris dans la plage	La valeur « 922337203 68547758071 » non valide a été fournie. Il doit être supérieur ou égal à 1 et inférieur ou égal à 922337203 6854775807.	
La colonne Barge-in n'accepte que la valeur booléenne	La valeur « test » non valide a été fournie. Il doit s'agir d'une valeur booléenne valide telle que « vrai » ou « faux ». Vous pouvez également utiliser « oui » et « non ».	Valeurs possibles : « Vrai », « vrai », « T », « Oui », « Oui », « Y », « 1 », « 1,0 », « faux », « faux », « F », « Non », « non », « N », « 0 », « 0,0 »
L'emplacement attendu, l'attribut de session et l'attribu t de demande doivent être séparés par un nombre égal à (=)	La valeur « SlotName: SlotValue » ne contient pas « = ». <value>Cette valeur doit être fournie sous forme de paire clé-valeur au format « <key>= ».	Par exemple : SlotName = SlotType
L'emplacement attendu, l'attribut de session et l'attribut de demande doivent avoir une paire clé-valeur	'=SlotValue' n'a pas de clé avant '='. <value>Cette valeur doit être fournie sous forme de paire clé-valeur au format « <key>= ».	Par exemple : SlotName = SlotType

Scénario	Message d'erreur	Remarques
Devis non valide à la fin	J'ai trouvé une citation incorrecte dans « Lenny's Burger ». Il commence par un guillemet « » mais ne se termine pas par le même guillemet.	Par exemple : `« Lenny's Burger », KFC`
Citation non valide au milieu	J'ai trouvé une citation incorrecte dans « Lenny's » Burger, KFC`. Il contient le caractère guillemet « » dans son contenu. Les valeurs contenant des guillemets simples doivent être placées entre guillemets doubles et vice-versa.	Correct Par exemple : `« Lenny's Burger », KFC`
Devis obligatoires	`key = Lenny's Burger` contient des guillemets simples ou doubles mais n'a pas été placé entre guillemet s. Les valeurs contenant des guillemets simples doivent être placées entre guillemets doubles et vice-versa.	
Clé dupliquée répétée dans la colonne	La clé « key1 » a été répétée sur deux colonnes : « Session Attribute 3 » et « Session Attribute 1 ».	

Scénario	Message d'erreur	Remarques
Format non valide dans Runtime hint	Clé non valide `BookFlig ht.Car. « `fourni pour Runtime Hints. Pour Runtime Hints, la clé doit être au format<in tentName>. <slotName>.	Si «. » doit être présent au milieu de la clé, le nom de l'intention et le nom de l'emplacement ne peuvent pas être extraits de cette clé. Exemples de formatage incorrect : "BookFlight«, ». BookFlight« Voiture », « BookFlight .Voiture ».
Nom d'intention non valide dans la clé d'indication d'exécution	Une intention `intent @name `non valide a été trouvée pour Runtime Hints. Vérifiez le nom de l'intention.	Vérification Regex : ^ ([0-9a-Za-Z] [_-] ?) +$
Nom d'emplacement non valide dans la clé d'indication d'exécution	Nom d'emplacement non valide détecté dans `Slot @Name `pour Runtime Hints. Vérifiez le nom du slot.	Régex : ^ ([0-9a-Za-Z] [_-] ?) + $Il ne doit pas commencer ou se terminer par un point (.)

Supprimer un ensemble de tests

Vous pouvez facilement supprimer un ensemble de tests de votre liste de tests.

Pour supprimer un ensemble de tests, procédez comme suit :

1. Accédez à la liste des ensembles de test dans le menu de gauche pour voir la liste des ensembles de test.
2. Dans la liste des ensembles de test, sélectionnez le set de test que vous souhaitez supprimer.
3. Accédez au menu déroulant Actions en haut à droite, puis choisissez Supprimer.
4. Un message confirme que le set de test a été supprimé.

Modifier les détails du set de test

Vous pouvez modifier le nom et les détails d'un ensemble de tests dans la liste des ensembles de tests. Le nom ou les détails peuvent être ajoutés ou mis à jour ultérieurement. Cependant, vous devrez mettre à jour votre ensemble de tests avant d'exécuter le test avec votre bot ou vos données de transcription.

Pour modifier les détails du set de test :

1. Accédez à la liste des ensembles de test dans le menu de gauche pour voir la liste des ensembles de test.
2. Dans la liste des ensembles de tests, cochez la case correspondant au set de test que vous souhaitez modifier.
3. Accédez au menu déroulant Actions en haut à droite, puis choisissez Modifier les détails.
4. Un message confirme que l'ensemble de tests a été correctement modifié.

Mettre à jour le kit de test

Vous pouvez mettre à jour, corriger, modifier ou supprimer des éléments de l'ensemble de tests afin d'optimiser vos résultats de référence ou de corriger d'autres erreurs susceptibles de s'être produites dans le kit de test

Vous pouvez télécharger un ensemble de tests et corriger les erreurs de validation avant de télécharger le kit de test corrigé. Voir Afficher les erreurs de validation des tests.

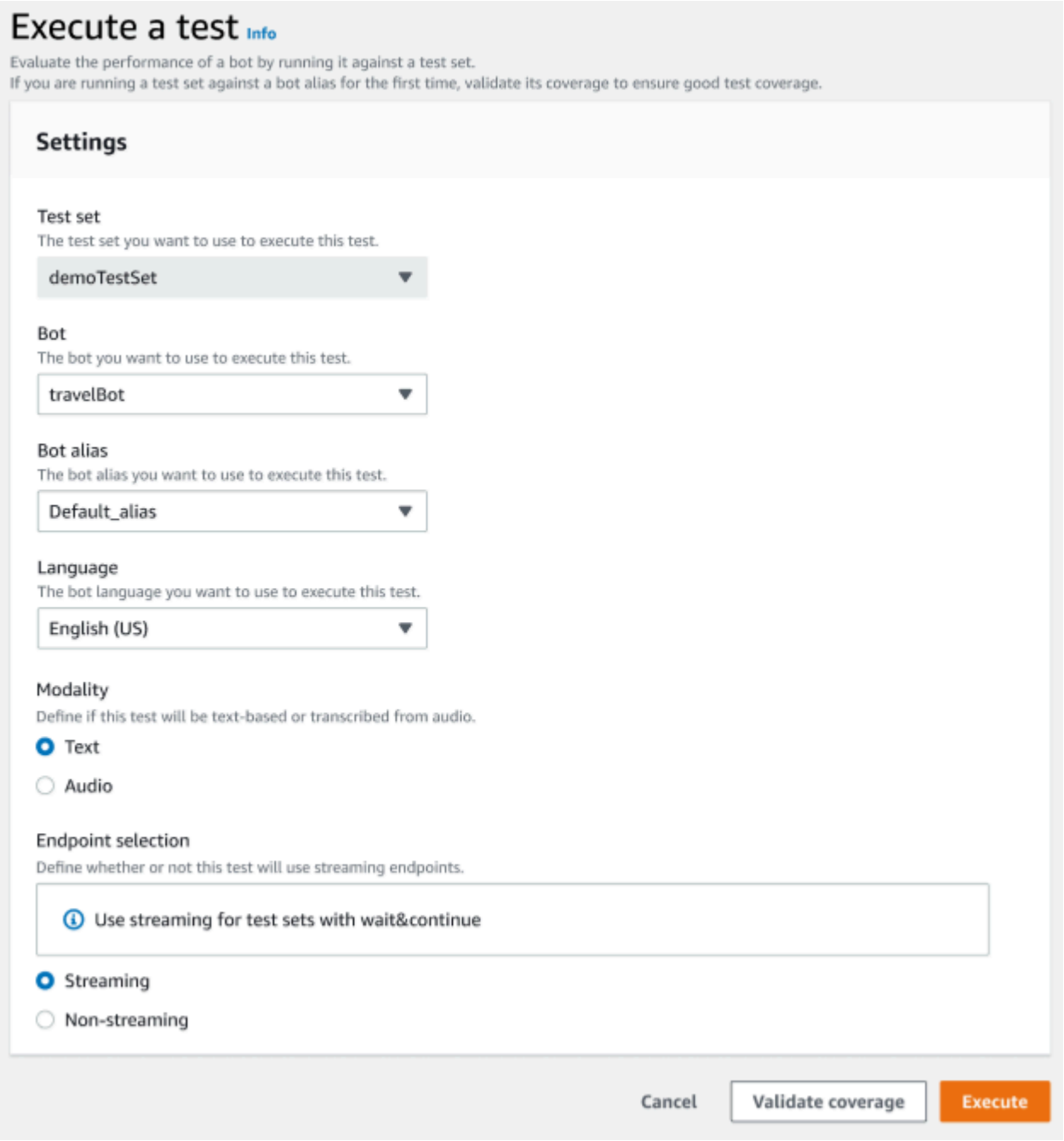
Pour mettre à jour un ensemble de tests :

1. Dans l'enregistrement du set de test, cliquez sur le bouton Mettre à jour le set de test en haut à droite.
2. Choisissez un fichier à télécharger depuis votre compte Amazon S3 ou chargez un fichier de test CSV depuis votre ordinateur. REMARQUE : La mise à jour d'un ensemble de tests remplacera les données existantes.
3. Sélectionnez le bouton Mettre à jour.
4. Un message confirme que l'ensemble de tests a été correctement mis à jour. REMARQUE : Cette opération peut prendre quelques minutes, en fonction de la complexité et de la taille du kit de test.

5. Un message confirme que le kit de test a été correctement mis à jour et le statut indique Prêt pour le test.

Exécuter un test

Pour exécuter un ensemble de tests, vous devez choisir le bot approprié pour exécuter le test par rapport à l'ensemble de tests. Vous pouvez choisir un bot depuis votre AWS compte dans le menu déroulant situé sous Test Set. Cette opération testera le bot que vous avez sélectionné par rapport à vos données de test validées afin de rapporter des mesures de performance par rapport aux données de référence du set de test.



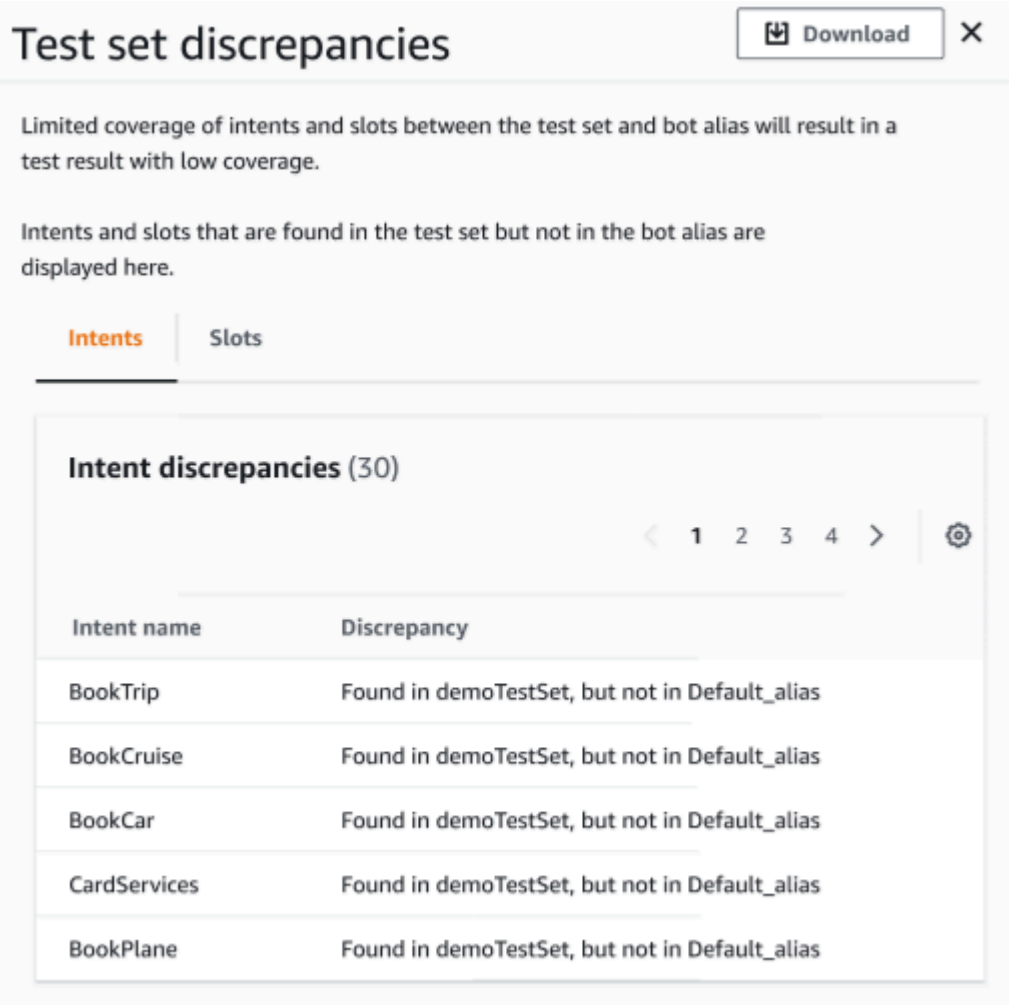
Pour exécuter un test dans le banc de test

1. Sur la page d'enregistrement du set de tests, choisissez Execute Test.
2. Sélectionnez le set de test que vous souhaitez utiliser dans le test.
3. Sélectionnez le nom du robot à utiliser dans le test dans le menu déroulant Bot.
4. Choisissez un alias de bot, le cas échéant, dans le menu déroulant des alias de bot.
5. Dans la sélection Langues, choisissez une version de l'anglais.

6. Sélectionnez Texte ou Audio pour le type de modalité.
7. Choisissez votre emplacement Amazon S3. (audio uniquement)
8. Sélectionnez le point de terminaison que vous avez sélectionné pour votre bot. (streaming uniquement)
9. Cliquez sur le bouton Valider la couverture pour confirmer que votre test est prêt à être exécuté. Si des erreurs sont présentes lors de l'étape de validation, passez en revue les paramètres précédents et apportez des corrections.
10. Sélectionnez Exécuter pour exécuter le test.
11. Un message confirme que le test a été exécuté avec succès.

Couverture du set de test

Une couverture limitée des intentions et des intervalles entre le set de test et le bot peut entraîner des mesures de performance attendues. Nous vous recommandons de vérifier la couverture du set de test avant d'exécuter le test.



Pour revoir la couverture de validation

1. Dans les enregistrements du set de test, cliquez sur le bouton Valider la couverture.
2. Le message indique qu'il valide la couverture entre le set de test et le bot sélectionné.
3. Une fois l'opération terminée, le message indique que la validation de la couverture est réussie.
4. Cliquez sur le bouton Afficher les détails en bas de la fenêtre.
5. Consultez les différences entre les ensembles de test en ce qui concerne les objectifs et les emplacements en choisissant l'onglet correspondant à chacun d'eux. Vous pouvez télécharger ces données au format CSV en cliquant sur le bouton Télécharger.
6. Passez en revue les résultats de validation pour les données de votre ensemble de tests, les intentions des robots et les créneaux. Identifiez les problèmes et apportez des modifications à l'architecture du set de test de votre bot afin d'améliorer les résultats. Téléchargez le jeu de tests modifié et un robot pour exécuter le test une fois que vous avez apporté des modifications au fichier CSV. REMARQUE : La couverture de validation s'applique à l'ensemble de tests et non

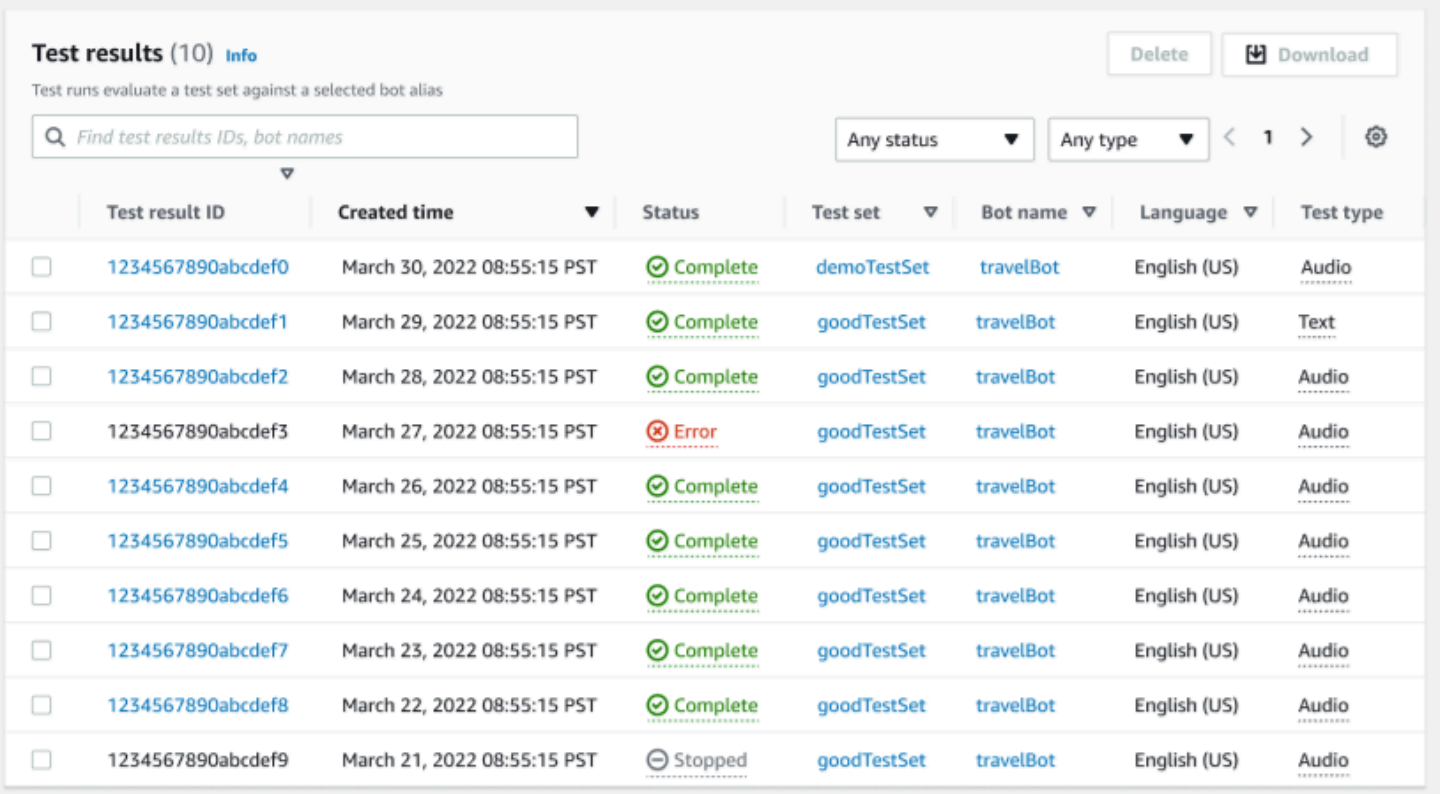
au bot. Les intentions présentes dans le bot mais absentes du kit de test ne seront pas prises en compte.

Afficher les résultats du test

Interprétez les résultats des tests depuis le banc de test pour déterminer dans quels cas la conversation entre votre bot et le client est susceptible d'échouer ou d'obliger le client à effectuer plusieurs tentatives pour atteindre son objectif.

En localisant ces problèmes dans les résultats de vos tests, vous pouvez optimiser les performances de votre robot en améliorant les performances d'intention en utilisant différentes données d'entraînement ou des énoncés plus cohérents avec les valeurs de transcription en temps réel du bot.

Vous pouvez obtenir une vue détaillée des intentions et des emplacements présentant des écarts de performances. Une fois que vous avez identifié les intentions ou les créneaux présentant des incohérences, vous pouvez approfondir et revoir les énoncés et le flux de conversation.



Pour consulter les résultats des tests :

1. Accédez à la liste des ensembles de tests dans le menu de gauche pour sélectionner l'option Résultats du test sous Test workbench. REMARQUE : Les résultats des tests indiquent que le statut est terminé s'ils sont réussis.
2. Sélectionnez l'ID du résultat du test pour les résultats du test que vous souhaitez consulter.

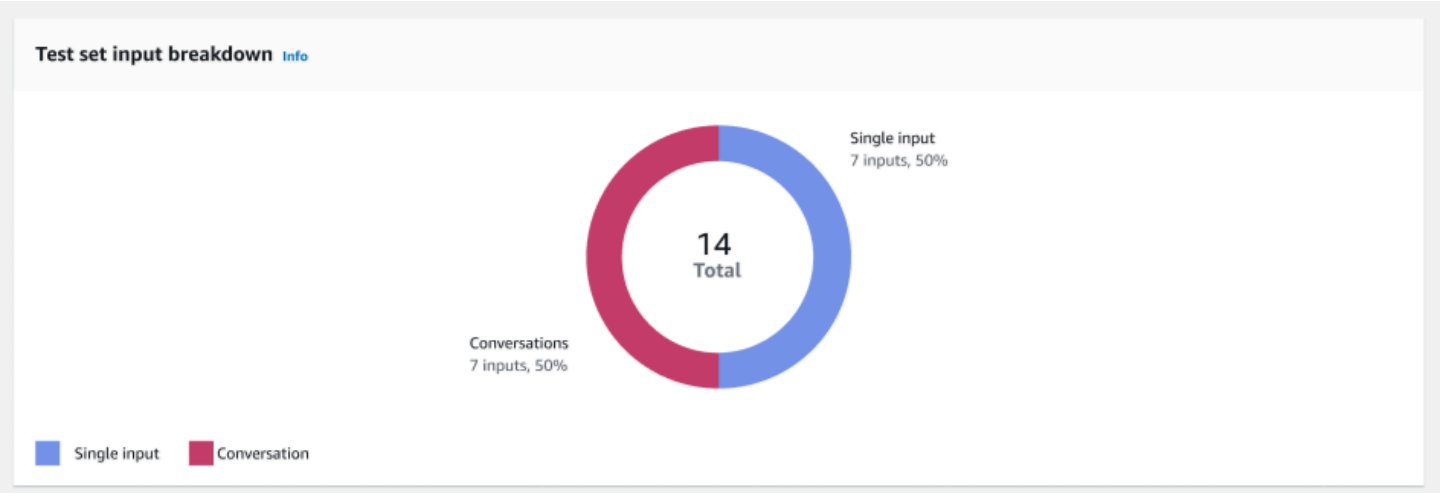
Détails des résultats des tests

Les résultats du test indiquent les détails du set de test, les intentions utilisées et les emplacements utilisés. Il fournit également la répartition globale des entrées du set de tests, y compris les résultats globaux, les résultats des conversations, l'intention et les résultats des créneaux.

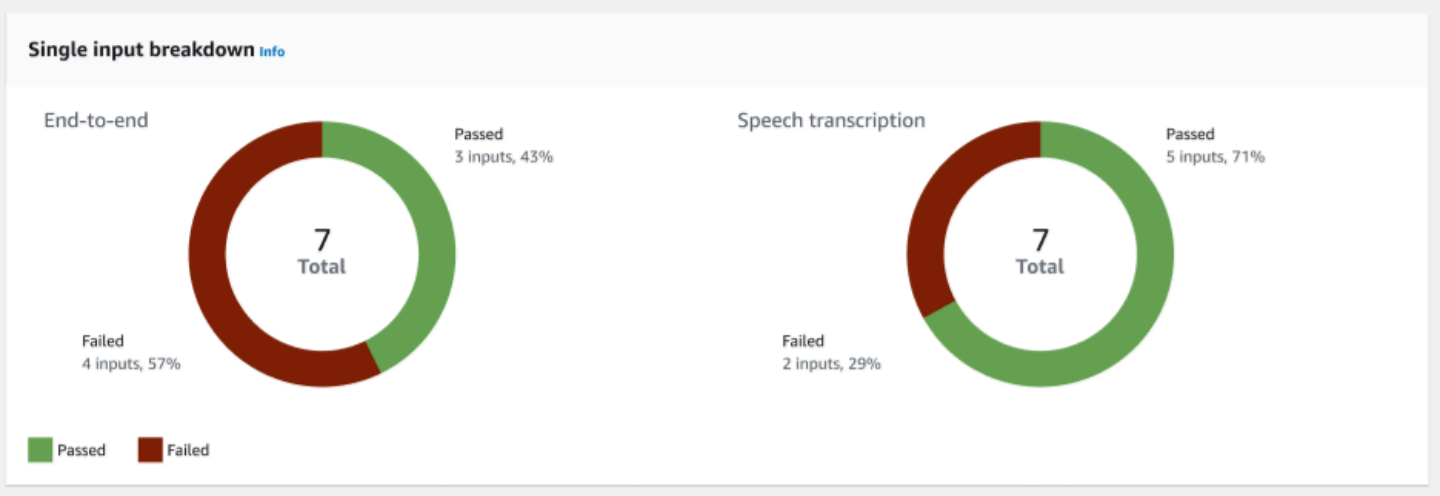
Les résultats des tests comprennent toutes les informations relatives aux tests, telles que :

- Métadonnées relatives aux détails du test
- Résultats globaux
- Résultats de la conversation
- Intention et résultats des créneaux
- Résultats détaillés

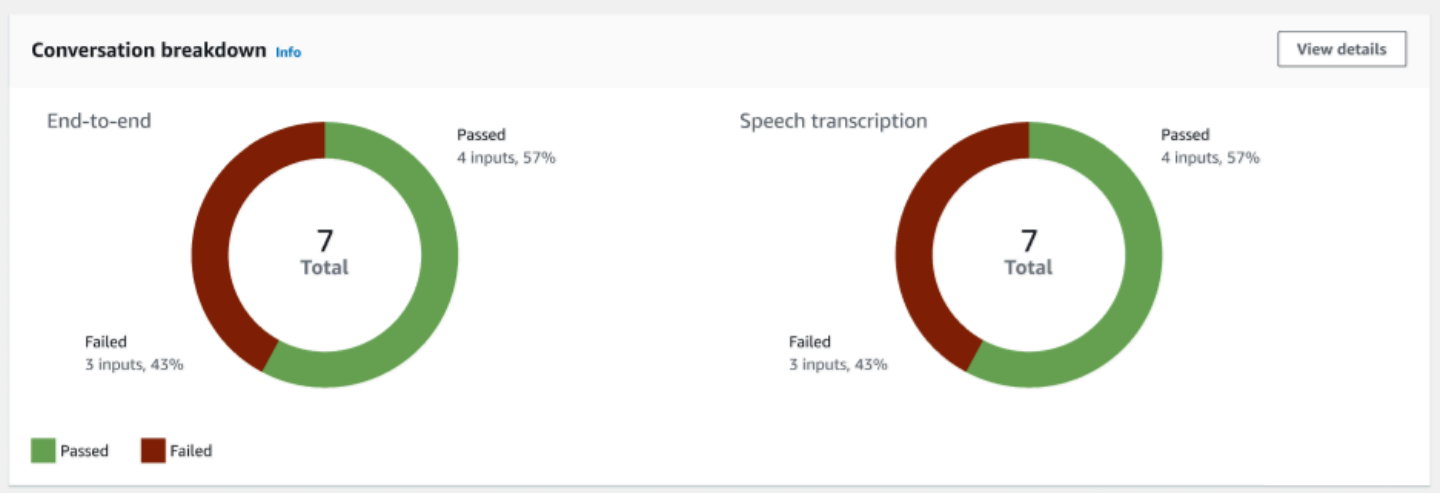
Onglet des résultats globaux :



Répartition des entrées du set de test — Ce graphique montre la répartition du nombre de conversations et d'énoncés à entrée unique dans le set de test.

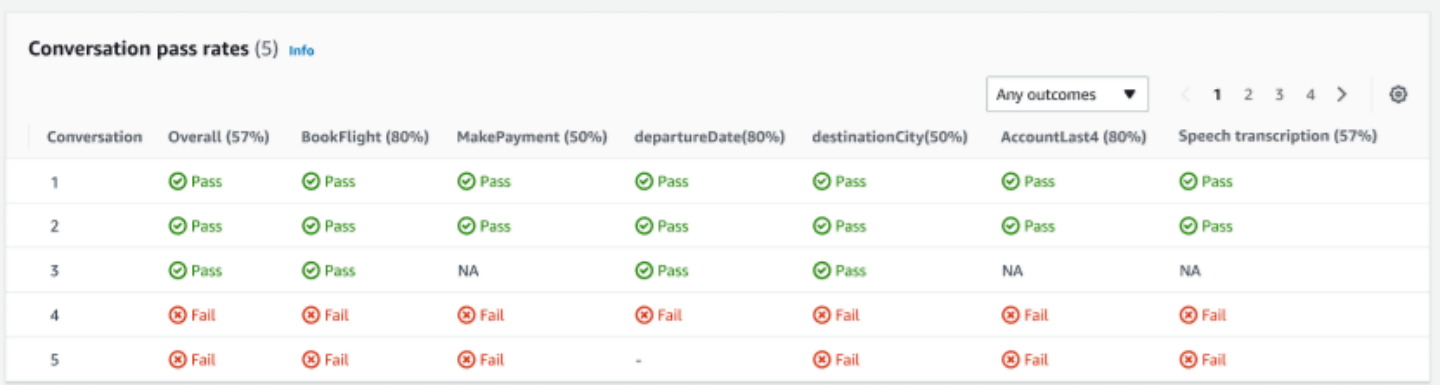


Répartition par entrée unique : affiche deux graphiques qui incluent end-to-end les conversations et les transcriptions vocales. Le nombre d'entrées réussies et échouées est indiqué sur chaque graphique. Remarque : le tableau de transcription vocale ne sera visible que pour le set de test audio.

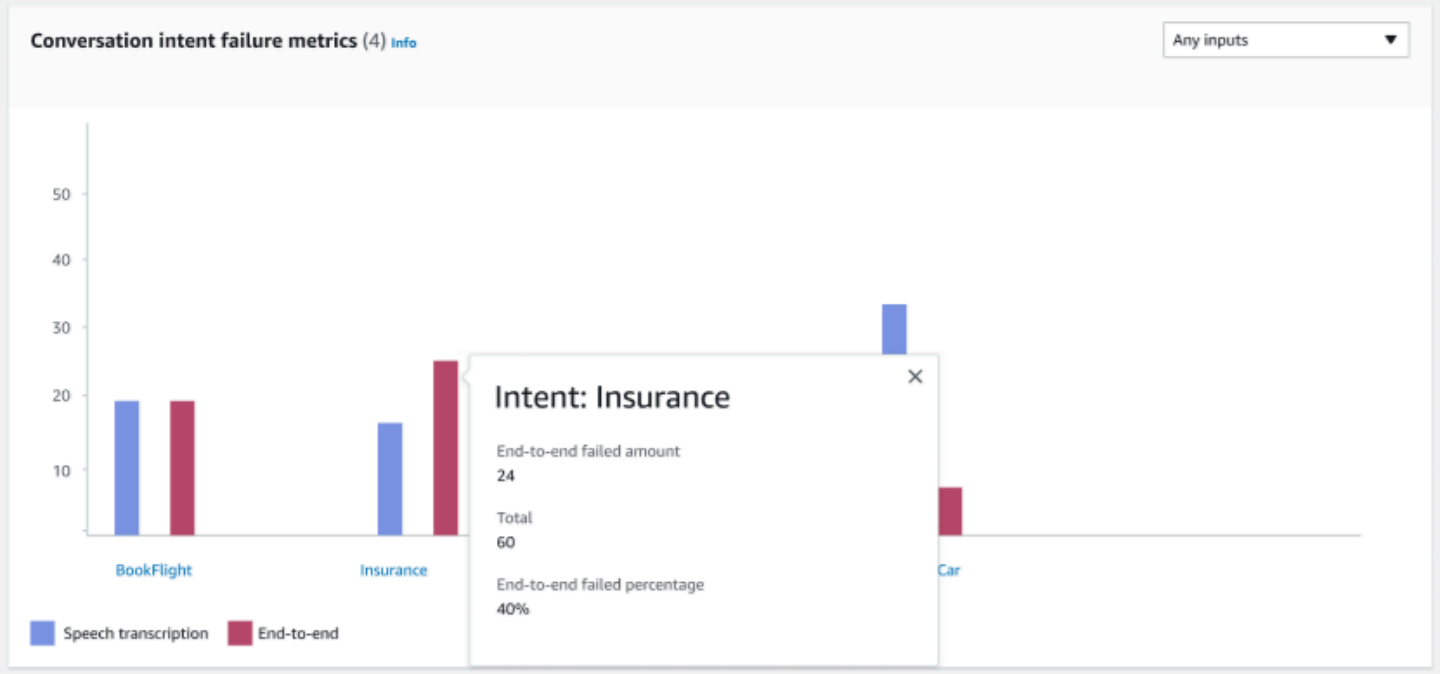


Répartition des conversations : affiche deux graphiques qui incluent end-to-end les conversations et les transcriptions vocales. Le nombre d'entrées réussies et échouées est indiqué sur chaque graphique. Remarque : le tableau de transcription vocale ne sera visible que pour le set de test audio.

Onglet des résultats de conversation :



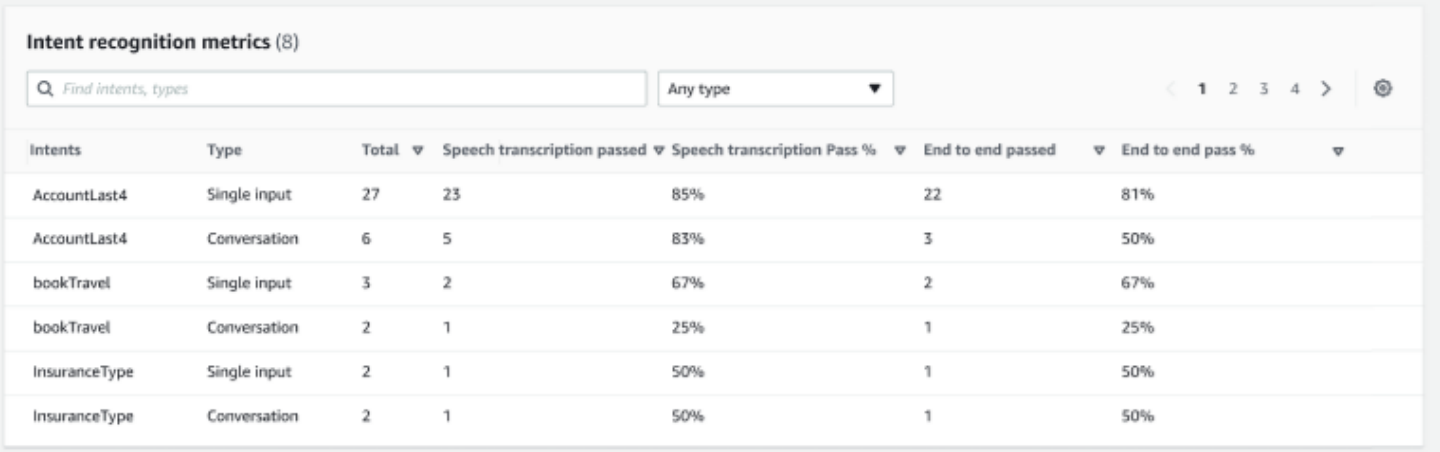
Taux de réussite des conversations : le tableau des taux de réussite des conversations est utilisé pour voir quelles intentions et quels créneaux sont utilisés dans chaque conversation du set de test. Vous pouvez visualiser où la conversation a échoué en examinant quelle intention ou quel créneau a échoué, ainsi que le pourcentage de réussite de chaque intention et de chaque créneau.



Indicateurs d'échec des intentions de conversation : cet indicateur indique les 5 intentions les moins performantes de l'ensemble de tests. Ce panneau affiche un graphique indiquant le pourcentage ou le nombre d'tentatives réussies ou échouées sur la base des journaux de conversation ou de la transcription du bot. Une intention réussie ne signifie pas que toute la conversation a été couronnée de succès. Ces mesures ne s'appliquent qu'à la valeur des intentions, quelle que soit l'intention antérieure ou ultérieure.

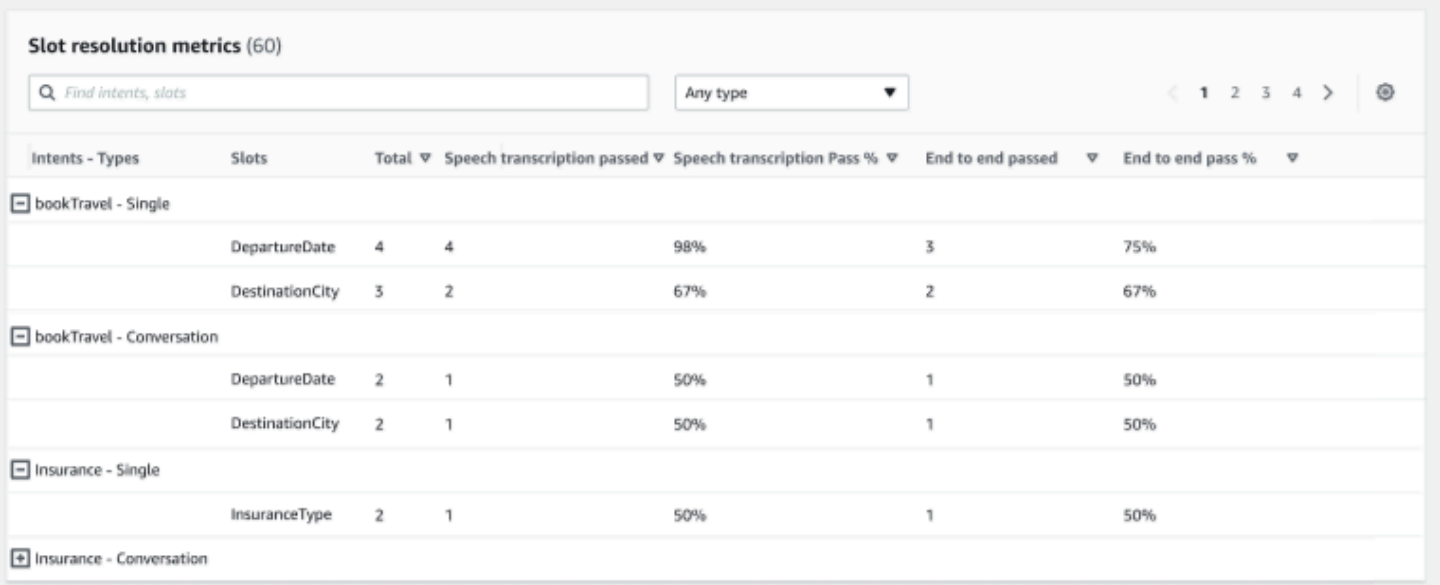
Indicateurs de défaillance des créneaux de conversation : cet indicateur indique les 5 emplacements les moins performants du set de test. Indique le taux de réussite pour chaque emplacement dans l'intention. Le graphique à barres montre à la fois la transcription vocale et les end-to-end conversations pour chaque créneau indiqué dans l'intention.

Onglet Intention et résultats des machines à sous :



Intents	Type	Total	Speech transcription passed	Speech transcription Pass %	End to end passed	End to end pass %
AccountLast4	Single input	27	23	85%	22	81%
AccountLast4	Conversation	6	5	83%	3	50%
bookTravel	Single input	3	2	67%	2	67%
bookTravel	Conversation	2	1	25%	1	25%
InsuranceType	Single input	2	1	50%	1	50%
InsuranceType	Conversation	2	1	50%	1	50%

Mesures de reconnaissance des intentions : affiche un tableau indiquant le nombre d'intentions reconnues avec succès. Affiche le taux de réussite de la transcription vocale et end-to-end des conversations.



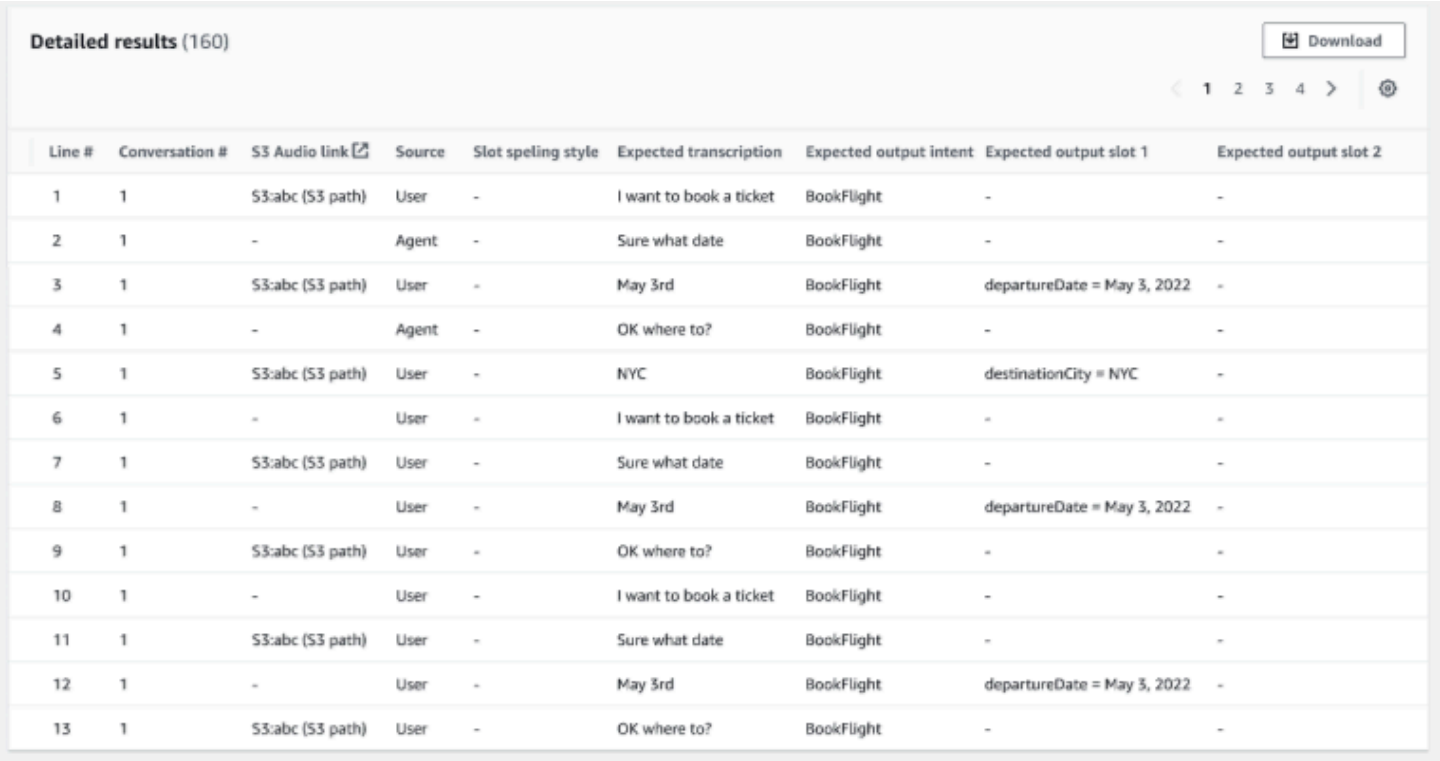
Slot resolution metrics (60)

Find intents, slots | Any type | 1 2 3 4

Intents - Types	Slots	Total	Speech transcription passed	Speech transcription Pass %	End to end passed	End to end pass %
bookTravel - Single						
	DepartureDate	4	4	98%	3	75%
	DestinationCity	3	2	67%	2	67%
bookTravel - Conversation						
	DepartureDate	2	1	50%	1	50%
	DestinationCity	2	1	50%	1	50%
Insurance - Single						
	InsuranceType	2	1	50%	1	50%
Insurance - Conversation						

Mesures de résolution des créneaux : affiche les intentions et les créneaux séparément, ainsi que le taux de réussite et d'échec de chaque intervalle pour chaque intention utilisée dans la conversation ou dans une entrée unique. Affiche le taux de réussite de la transcription vocale et end-to-end des conversations.

Onglet des résultats détaillés :



Detailed results (160)

Download | 1 2 3 4

Line #	Conversation #	S3 Audio link	Source	Slot speling style	Expected transcription	Expected output intent	Expected output slot 1	Expected output slot 2
1	1	S3:abc (S3 path)	User	-	I want to book a ticket	BookFlight	-	-
2	1	-	Agent	-	Sure what date	BookFlight	-	-
3	1	S3:abc (S3 path)	User	-	May 3rd	BookFlight	departureDate = May 3, 2022	-
4	1	-	Agent	-	OK where to?	BookFlight	-	-
5	1	S3:abc (S3 path)	User	-	NYC	BookFlight	destinationCity = NYC	-
6	1	-	User	-	I want to book a ticket	BookFlight	-	-
7	1	S3:abc (S3 path)	User	-	Sure what date	BookFlight	-	-
8	1	-	User	-	May 3rd	BookFlight	departureDate = May 3, 2022	-
9	1	S3:abc (S3 path)	User	-	OK where to?	BookFlight	-	-
10	1	-	User	-	I want to book a ticket	BookFlight	-	-
11	1	S3:abc (S3 path)	User	-	Sure what date	BookFlight	-	-
12	1	-	User	-	May 3rd	BookFlight	departureDate = May 3, 2022	-
13	1	S3:abc (S3 path)	User	-	OK where to?	BookFlight	-	-

Résultats détaillés — Affiche un tableau détaillé sur le journal des conversations avec les énoncés de l'utilisateur et de l'agent, ainsi que le résultat attendu et la transcription attendue pour chaque intervalle. Vous pouvez télécharger ce rapport en cliquant sur le bouton Télécharger.

Le tableau suivant répertorie les messages d'erreur liés à l'échec des résultats avec des scénarios.

Scénario	Message d’erreur	Action
Incompatibilité d'intention	BookFlight Intention attendue, mais c'était BookHotel une intention.	Ignorer les autres tournants de la conversation
Incompatibilité entre les machines à sous Elicitation	Le créneau DepartureDate attendu devait être obtenu, mais il s'agissait de CabinType .	Ignorer les autres tournants de la conversation
Incompatibilité de la valeur du slot	Incompatibilité entre la valeur attendue et la valeur réelle de l'emplacement.	Poursuivez les conversations à d'autres reprises
ack-to-back L'invite de l'agent B est manquante	On s'attendait à ce que le bot renvoie une invite à l'agent à ce tour, mais celle-ci n'a pas été reçue.	Ignorer les autres tournants de la conversation
Incompatibilité de transcription	La transcription attendue ne correspond pas à la transcrip tion réelle.	Poursuivez les conversations à d'autres reprises
Emplacement optionnel non obtenu	On s'attend à obtenir le slot CabinType au prochain tour, mais l'intention actuelle a été remplie avant cela.	Ignorer les autres tournants de la conversation
Emplacement non reconnu	Le créneau Expected DepartureDate n'a pas été reconnu lors de ce tour.	Ignorer les autres tournants de la conversation

Scénario	Message d’erreur	Action
Demande d' back-to-back agent supplémentaire	Je m'attendais à un tour d'utilisateur mais c'était une demande de l'agent	Ignorer les autres tournants de la conversation

Streaming vers un bot Amazon Lex V2

Vous pouvez utiliser l'API de streaming Amazon Lex V2 pour démarrer un flux bidirectionnel entre un bot Amazon Lex V2 et votre application. Le démarrage d'un stream permet au bot de gérer la conversation entre le bot et l'utilisateur. Le bot répond aux entrées de l'utilisateur sans que vous n'écriviez de code pour gérer les réponses de l'utilisateur. Le bot peut :

- Gérez les interruptions de l'utilisateur pendant la lecture d'une invite. Pour plus d'informations, veuillez consulter Permettre à votre bot d'être interrompu par votre utilisateur.
- Attendez que l'utilisateur fournisse une entrée. Par exemple, le bot peut attendre que l'utilisateur collecte les informations de carte de crédit. Pour plus d'informations, veuillez consulter Permettre au bot d'attendre que l'utilisateur fournisse plus d'informations.
- Transférez à la fois une entrée audio et une entrée audio à double tonalité et multifréquence (DTMF) dans le même flux.
- Gérez mieux les pauses dans les saisies utilisateur que si vous gériez la conversation depuis votre application.

Non seulement le bot Amazon Lex V2 répond aux données envoyées depuis votre application, mais il envoie également des informations sur l'état de la conversation à votre application. Vous pouvez utiliser ces informations pour modifier la façon dont votre application répond aux clients.

Le bot Amazon Lex V2 surveille également la connexion entre le bot et votre application. Il peut déterminer si le délai de connexion a expiré.

Pour utiliser l'API afin de démarrer un flux vers un bot Amazon Lex V2, consultezLancer un stream vers un bot.

Lorsque vous commencez à diffuser du contenu vers un bot Amazon Lex V2 depuis votre application, vous pouvez configurer le bot pour qu'il accepte les entrées audio ou texte de l'utilisateur. Vous pouvez également choisir si l'utilisateur reçoit du son ou du texte en réponse à sa saisie.

Si vous avez configuré le bot Amazon Lex V2 pour accepter les entrées audio de l'utilisateur, il ne peut pas accepter de saisie de texte. Si vous avez configuré le robot pour qu'il accepte la saisie de texte, l'utilisateur ne peut utiliser que du texte écrit pour communiquer avec lui.

Lorsqu'un robot Amazon Lex V2 reçoit une entrée audio en streaming, il détermine quand un utilisateur commence à parler et quand il arrête de parler. Il gère les pauses ou les interruptions de

l'utilisateur. Il peut également prendre en charge une entrée DTMF (multifréquence bicolore) et une entrée vocale dans le même flux. Cela permet à l'utilisateur d'interagir plus naturellement avec le bot. Vous pouvez présenter aux utilisateurs des messages de bienvenue et des instructions. Vous pouvez également autoriser les utilisateurs à interrompre ces messages et ces invites.

Lorsque vous démarrez un flux bidirectionnel, Amazon Lex V2 utilise le protocole HTTP/2. Votre application et le bot échangent des données dans un flux unique sous la forme d'une série d'événements. Un événement peut être l'un des suivants :

- Entrée de texte, audio ou DTMF provenant de l'utilisateur.
- Signaux de l'application vers le bot Amazon Lex V2. Il s'agit notamment d'une indication que la lecture audio d'un message est terminée ou que l'utilisateur s'est déconnecté de la session.

Pour plus d'informations sur les événements, consultez Lancer un stream vers un bot. Pour plus d'informations sur la façon de coder les événements, consultezCodage du flux d'événements.

Rubriques

Lancer un stream vers un bot

Vous utilisez cette StartConversationopération pour démarrer un flux entre l'utilisateur et le bot Amazon Lex V2 dans votre application. La`POST` demande de l'application établit une connexion entre votre application et le bot Amazon Lex V2. Cela permet à votre application et au bot de commencer à échanger des informations par le biais d'événements.

L'`StartConversation`opération n'est prise en charge que dans les SDK suivants :

- AWS SDK pour C++
- AWS SDK for Java V2
- Kit SDK AWS pour JavaScript v3

- AWSSDK pour Ruby V3

Le premier événement que votre application doit envoyer au bot Amazon Lex V2 est un ConfigurationEvent. Cet événement inclut des informations telles que le format du type de réponse. Les paramètres que vous pouvez utiliser dans un événement de configuration sont les suivants :

- responseContentType— Détermine si le bot répond à la saisie de l'utilisateur par du texte ou de la parole.
- SessionState : informations relatives à la session de streaming avec le bot, telles que l'intention prédéterminée ou l'état de dialogue.
- WelcomeMessages — Spécifie les messages de bienvenue qui s'affichent à l'utilisateur au début de sa conversation avec un robot. Ces messages sont lus avant que l'utilisateur ne fournisse de données. Pour activer un message de bienvenue, vous devez également spécifier les valeurs pour les`dialogAction` paramètres`sessionState` et.
- DisablePlayback — Détermine si le robot doit attendre un signal du client avant de commencer à écouter les entrées de l'appelant. Par défaut, la lecture est activée. La valeur de ce champ est donc`false`.
- requestAttributes — Fournit des informations supplémentaires concernant la demande.

Pour plus d'informations sur la manière de spécifier les valeurs des paramètres précédents, consultez le type de ConfigurationEventdonnées de l'StartConversationopération.

Chaque flux entre un bot et votre application ne peut avoir qu'un seul événement de configuration. Une fois que votre application a envoyé un événement de configuration, le bot peut recevoir des communications supplémentaires en provenance de votre application.

Si vous avez indiqué que votre utilisateur utilise le son pour communiquer avec le robot Amazon Lex V2, votre application peut envoyer les événements suivants au robot au cours de cette conversation :

- AudioInputEvent— Contient un morceau audio dont la taille maximale est de 320 octets. Votre application doit utiliser plusieurs événements d'entrée audio pour envoyer un message du serveur au bot. Chaque événement d'entrée audio du flux doit avoir le même format audio.
- DTMFInputEvent — Envoie une entrée DTMF au bot. Chaque pression sur une touche DTMF correspond à un événement unique.
- PlaybackCompletionEvent— Informe le serveur qu'une réponse provenant de la saisie de l'utilisateur lui a été lue. Vous devez utiliser un événement de fin de lecture si vous envoyez une

réponse audio à l'utilisateur. Si `disablePlayback` l'un de vos événements de configuration `setrue` produit, vous ne pouvez pas utiliser cette fonctionnalité.

- DisconnectionEvent— Informe le bot que l'utilisateur s'est déconnecté de la conversation.

Si vous avez indiqué que l'utilisateur utilise du texte pour communiquer avec le robot, votre application peut envoyer les événements suivants au bot au cours de cette conversation :

- TextInputEvent— Texte envoyé depuis votre application au bot. Un événement de saisie de texte peut contenir jusqu'à 512 caractères.
- PlaybackCompletionEvent— Informe le serveur qu'une réponse provenant de la saisie de l'utilisateur lui a été lue. Vous devez utiliser cet événement si vous rediffusez du son à l'utilisateur. Si `disablePlayback` l'un de vos événements de configuration `setrue` produit, vous ne pouvez pas utiliser cette fonctionnalité.
- DisconnectionEvent— Informe le bot que l'utilisateur s'est déconnecté de la conversation.

Vous devez encoder chaque événement que vous envoyez à un bot Amazon Lex V2 dans le format approprié. Pour plus d'informations, veuillez consulter Codage du flux d'événements.

Chaque événement possède un ID d'événement. Pour vous aider à résoudre les problèmes susceptibles de survenir dans le flux, attribuez un identifiant d'événement unique à chaque événement d'entrée. Vous pouvez ensuite résoudre tout échec de traitement à l'aide du bot.

Amazon Lex V2 utilise également des horodatages pour chaque événement. Vous pouvez utiliser ces horodatages en plus de l'ID d'événement pour résoudre tout problème de transmission réseau.

Au cours de la conversation entre l'utilisateur et le robot Amazon Lex V2, le bot peut envoyer les événements sortants suivants en réponse à l'utilisateur :

- IntentResultEvent— Contient l'intention qu'Amazon Lex V2 a déterminée à partir de l'énoncé de l'utilisateur. Chaque événement de résultat interne inclut :
  - InputMode — Type d'énoncé utilisateur. Les valeurs valides sont `Speech`, `DTMF` ou `Text`.
  - interprétations : interprétations qu'Amazon Lex V2 détermine à partir de l'énoncé de l'utilisateur.
  - requestAttributes — Si vous n'avez pas modifié les attributs de la demande à l'aide d'une fonction lambda, il s'agit des mêmes attributs que ceux qui ont été transmis au début de la conversation.
  - SessionId — Identifiant de session utilisé pour la conversation.

  - SessionState : état de la session de l'utilisateur avec Amazon Lex V2.
- TranscriptEvent— Si l'utilisateur fournit une entrée à votre application, cet événement contient la transcription de la déclaration de l'utilisateur au bot. Votre application ne reçoit pas de message`TranscriptEvent` s'il n'y a aucune entrée utilisateur.

  La valeur de l'événement de transcription envoyé à votre application varie selon que vous avez sélectionné le mode audio (voix et DMTF) ou le mode texte comme mode de conversation :

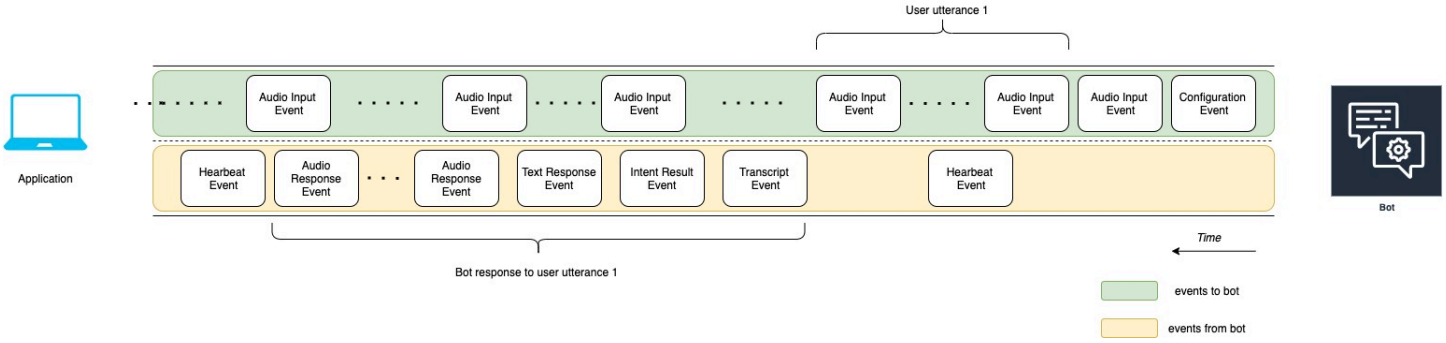
  - Transcription de la saisie vocale : si l'utilisateur parle avec le robot, l'événement de transcription est la transcription du son de l'utilisateur. Il s'agit d'une transcription de tout le discours entre le moment où l'utilisateur commence à parler et le moment où il termine de parler.
  - Transcription de la saisie DTMF : si l'utilisateur tape sur un clavier, l'événement de transcription contient tous les chiffres sur lesquels l'utilisateur a appuyé lors de sa saisie.
  - Transcription de la saisie de texte : si l'utilisateur saisit du texte, l'événement de transcription contient tout le texte saisi par l'utilisateur.
- TextResponseEvent— Contient la réponse du bot au format texte. Une réponse textuelle est renvoyée par défaut. Si vous avez configuré Amazon Lex V2 pour renvoyer une réponse audio, ce texte est utilisé pour générer une réponse audio. Chaque événement de réponse textuelle contient un tableau d'objets de message que le bot renvoie à l'utilisateur.
- AudioResponseEvent— Contient la réponse audio synthétisée à partir du texte généré dans le`TextResponseEvent`. Pour recevoir des événements de réponse audio, vous devez configurer Amazon Lex V2 afin de fournir une réponse audio. Tous les événements de réponse audio ont le même format audio. Chaque événement contient des segments audio ne dépassant pas 100 octets. Amazon Lex V2 envoie un segment audio vide avec le`bytes` champ défini`null` pour indiquer la fin de l'événement de réponse audio à votre application.
- PlaybackInterruptionEvent— Lorsqu'un utilisateur interrompt une réponse que le bot a envoyée à votre application, Amazon Lex V2 déclenche cet événement pour arrêter la lecture de la réponse.
- HeartbeatEvent— Amazon Lex V2 renvoie cet événement régulièrement pour empêcher la connexion entre votre application et le bot d'atteindre le délai imparti.

Séquence temporelle des événements pour une conversation audio

Les diagrammes suivants montrent une conversation audio en streaming entre un utilisateur et un robot Amazon Lex V2. L'application diffuse du son en continu vers le bot, qui recherche les entrées de l'utilisateur à partir de l'audio. Dans cet exemple, l'utilisateur et le bot utilisent la parole pour

communiquer. Chaque diagramme correspond à un énoncé utilisateur et à la réponse du bot à cet énoncé.

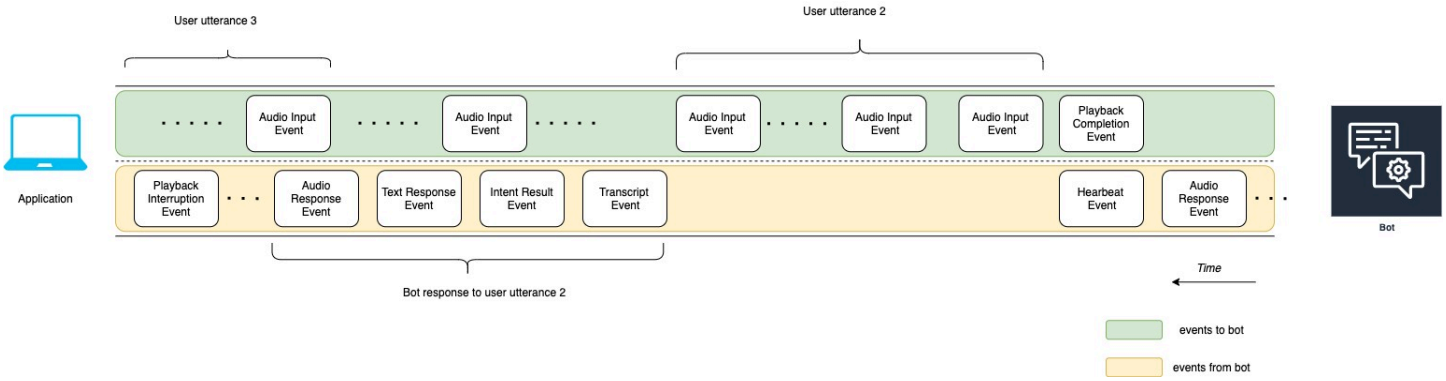
Le schéma suivant montre le début d'une conversation entre l'application et le bot. Le flux commence à l'heure zéro (t0).



La liste suivante décrit les événements du schéma précédent.

- t0 : L'application envoie un événement de configuration au bot pour démarrer le flux.
- t1 : L'application diffuse des données audio. Ces données sont réparties en une série d'événements d'entrée provenant de l'application.
- t2 : Pour l'énoncé utilisateur 1, le bot détecte un événement d'entrée audio lorsque l'utilisateur commence à parler.
- t2 : Pendant que l'utilisateur parle, le bot envoie un événement de battement de cœur pour maintenir la connexion. Il envoie ces événements par intermittence pour s'assurer que la connexion ne s'interrompt pas.
- t3 : Le bot détecte la fin de la déclaration de l'utilisateur.
- t4 : Le bot renvoie à l'application un événement de transcription contenant une transcription du discours de l'utilisateur. C'est le début de la réponse du bot à l'énoncé 1 de l'utilisateur.
- t5 : Le bot envoie un événement Intent Result pour indiquer l'action que l'utilisateur souhaite effectuer.
- t6 : Le bot commence à fournir sa réponse sous forme de texte dans un événement de réponse textuelle.
- t7 : Le bot envoie une série d'événements de réponse audio à l'application pour qu'elle soit diffusée à la place de l'utilisateur.
- t8 : Le bot envoie un autre événement de rythme cardiaque pour maintenir la connexion par intermittence.

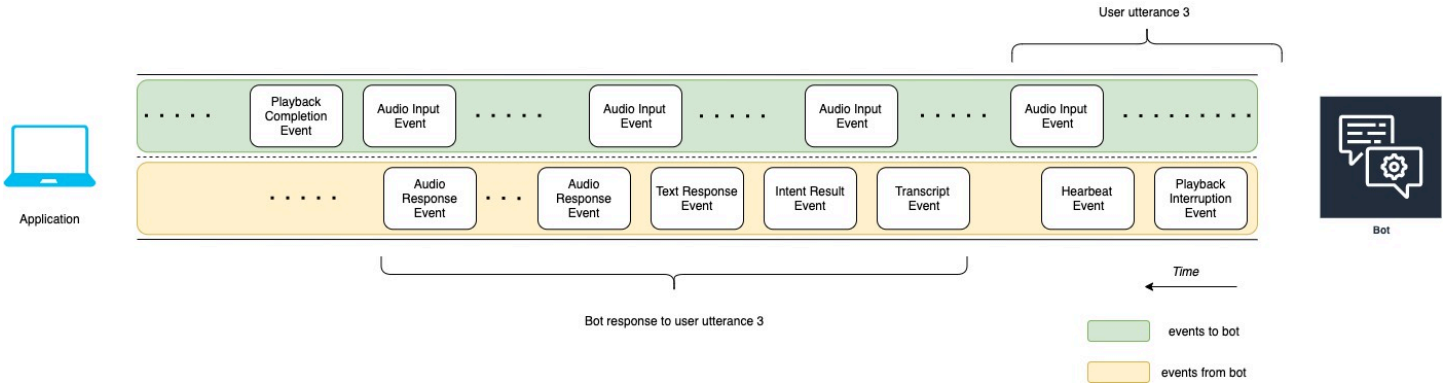
Le schéma suivant est une continuation du schéma précédent. Il montre que l'application envoie un événement de fin de lecture au robot pour indiquer qu'elle a arrêté de lire la réponse audio pour l'utilisateur. L'application lit la réponse du bot à l'énoncé 1 de l'utilisateur. L'utilisateur répond à la réponse du bot à l'énoncé utilisateur 1 par l'énoncé utilisateur 2.



La liste suivante décrit les événements du schéma précédent :

- t10 : L'application envoie un événement de fin de lecture pour indiquer qu'elle a fini de lire le message du bot à l'utilisateur.
- t11 : L'application renvoie la réponse de l'utilisateur au bot sous la forme d'énoncé utilisateur 2.
- t12 : Pour la réponse du bot à l'énoncé 2 de l'utilisateur, le robot attend que l'utilisateur arrête de parler, puis commence à fournir une réponse audio.
- t13 : Pendant que le bot envoie la réponse Bot à l'énoncé utilisateur 2 à l'application, le bot détecte le début de l'énoncé utilisateur 3. Le bot arrête la réponse du bot à l'énoncé 2 de l'utilisateur et envoie un événement d'interruption de lecture.
- t14 : Le bot envoie un événement d'interruption de lecture à l'application pour signaler que l'utilisateur a interrompu l'invite.

Le schéma suivant montre la réponse du bot à l'énoncé de l'utilisateur 3 et indique que la conversation se poursuit une fois que le robot a répondu à l'énoncé de l'utilisateur.



Utilisation de l'API pour démarrer une conversation en streaming

Lorsque vous lancez un stream vers un bot Amazon Lex V2, vous effectuez les tâches suivantes :

1. Créez une connexion initiale au serveur.
2. Configurez les informations d'identification de sécurité et les détails du bot. Les détails du bot indiquent si le bot accepte le DTMF et les entrées audio, ou la saisie de texte.
3. Envoyez des événements au serveur. Ces événements sont des données textuelles ou audio provenant de l'utilisateur.
4. Traitez les événements envoyés depuis le serveur. Au cours de cette étape, vous déterminez si la sortie du bot est présentée à l'utilisateur sous forme de texte ou de parole.

Les exemples de code suivants initialisent une conversation en streaming avec un bot Amazon Lex V2 et votre machine locale. Vous pouvez modifier le code afin de répondre à vos besoins.

Le code suivant est un exemple de demande utilisant leAWS SDK for Java pour démarrer la connexion à un robot et configurer les détails et les informations d'identification du bot.

```
package com.lex.streaming.sample;

import software.amazon.awssdk.auth.credentials.AwsBasicCredentials;
import software.amazon.awssdk.auth.credentials.AwsCredentialsProvider;
import software.amazon.awssdk.auth.credentials.StaticCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lexruntimev2.LexRuntimeV2AsyncClient;
import software.amazon.awssdk.services.lexruntimev2.model.ConversationMode;
import software.amazon.awssdk.services.lexruntimev2.model.StartConversationRequest;

import java.net.URISyntaxException;
```

```
import java.util.UUID;
import java.util.concurrent.CompletableFuture;

/**
 * The following code creates a connection with the Amazon Lex bot and configures the
 bot details and credentials.
 * Prerequisite: To use this example, you must be familiar with the Reactive streams
 programming model.
 * For more information, see
 * https://github.com/reactive-streams/reactive-streams-jvm.
 * This example uses AWS SDK for Java for Amazon Lex V2.
 * <p>
 * The following sample application interacts with an Amazon Lex bot with the streaming
 API. It uses the Audio
 * conversation mode to return audio responses to the user's input.
 * <p>
 * The code in this example accomplishes the following:
 * <p>
 * 1. Configure details about the conversation between the user and the Amazon Lex bot.
 These details include the conversation mode and the specific bot the user is speaking
 with.
 * 2. Create an events publisher that passes the audio events to the Amazon Lex bot
 after you establish the connection. The code we provide in this example tells your
 computer to pick up the audio from
 * your microphone and send that audio data to Amazon Lex.
 * 3. Create a response handler that handles the audio responses from the Amazon Lex
 bot and plays back the audio to you.
 */
public class LexBidirectionalStreamingExample {

    public static void main(String[] args) throws URISyntaxException,
 InterruptedException {
        String botId = "";
        String botAliasId = "";
        String localeId = "";
        String accessKey = "";
        String secretKey = "";
        String sessionId = UUID.randomUUID().toString();
        Region region = Region.region_name; // Choose an AWS Region where the Amazon
 Lex Streaming API is available.

        AwsCredentialsProvider awsCredentialsProvider = StaticCredentialsProvider
                .create(AwsBasicCredentials.create(accessKey, secretKey));
```

```
        // Create a new SDK client. You need to use an asynchronous client.
        System.out.println("step 1: creating a new Lex SDK client");
        LexRuntimeV2AsyncClient lexRuntimeServiceClient =
LexRuntimeV2AsyncClient.builder()
                .region(region)
                .credentialsProvider(awsCredentialsProvider)
                .build();


        // Configure the bot, alias and locale that you'll use to have a conversation.
        System.out.println("step 2: configuring bot details");
        StartConversationRequest.Builder startConversationRequestBuilder =
StartConversationRequest.builder()
                .botId(botId)
                .botAliasId(botAliasId)
                .localeId(localeId);

        // Configure the conversation mode of the bot. By default, the
        // conversation mode is audio.
        System.out.println("step 3: choosing conversation mode");
        startConversationRequestBuilder =
startConversationRequestBuilder.conversationMode(ConversationMode.AUDIO);

        // Assign a unique identifier for the conversation.
        System.out.println("step 4: choosing a unique conversation identifier");
        startConversationRequestBuilder =
startConversationRequestBuilder.sessionId(sessionId);

        // Start the initial request.
        StartConversationRequest startConversationRequest =
startConversationRequestBuilder.build();

        // Create a stream of audio data to the Amazon Lex bot. The stream will start
after the connection is established with the bot.
        EventsPublisher eventsPublisher = new EventsPublisher();

        // Create a class to handle responses from bot. After the server processes the
user data you've streamed, the server responds
        // on another stream.
        BotResponseHandler botResponseHandler = new
BotResponseHandler(eventsPublisher);

        // Start a connection and pass in the publisher that streams the audio and
process the responses from the bot.
```

```
        System.out.println("step 5: starting the conversation ...");
        CompletableFuture<Void> conversation =
 lexRuntimeServiceClient.startConversation(
                startConversationRequest,
                eventsPublisher,
                botResponseHandler);

        // Wait until the conversation finishes. The conversation finishes if the
 dialog state reaches the "Closed" state.
        // The client stops the connection. If an exception occurs during the
 conversation, the
        // client sends a disconnection event.
        conversation.whenComplete((result, exception) -> {
            if (exception != null) {
                eventsPublisher.disconnect();
            }
        });

        // The conversation finishes when the dialog state is closed and last prompt
 has been played.
        while (!botResponseHandler.isConversationComplete()) {
            Thread.sleep(100);
        }

        // Randomly sleep for 100 milliseconds to prevent JVM from exiting.
        // You won't need this in your production code because your JVM is
        // likely to always run.
        // When the conversation finishes, the following code block stops publishing
 more data and informs the Amazon Lex bot that there is no more data to send.
        if (botResponseHandler.isConversationComplete()) {
            System.out.println("conversation is complete.");
            eventsPublisher.stop();
        }
    }
}
```

Le code suivant est un exemple de demande utilisant leAWS SDK for Java pour envoyer des événements au bot. Le code de cet exemple utilise le microphone de votre ordinateur pour envoyer des événements audio.

```
package com.lex.streaming.sample;

import org.reactivestreams.Publisher;
import org.reactivestreams.Subscriber;
import
 software.amazon.awssdk.services.lexruntimev2.model.StartConversationRequestEventStream;

/**
 * You use the Events publisher to send events to the Amazon Lex bot. When you
 establish a connection, the bot uses the
 * subscribe() method and enables the events publisher starts sending events to
 * your computer. The bot uses the "request" method of the subscription to make more
 requests. For more information on the request method, see https://github.com/reactive-
streams/reactive-streams-jvm.
 */
public class EventsPublisher implements Publisher<StartConversationRequestEventStream>
 {

    private AudioEventsSubscription audioEventsSubscription;

    @Override
    public void subscribe(Subscriber<? super StartConversationRequestEventStream>
 subscriber) {
        if (audioEventsSubscription == null) {

            audioEventsSubscription = new AudioEventsSubscription(subscriber);
            subscriber.onSubscribe(audioEventsSubscription);

        } else {
            throw new IllegalStateException("received unexpected subscription
 request");
        }
    }

    public void disconnect() {
        if (audioEventsSubscription != null) {
            audioEventsSubscription.disconnect();
        }
    }

    public void stop() {
        if (audioEventsSubscription != null) {
            audioEventsSubscription.stop();
        }
```

```
    }

    public void playbackFinished() {
        if (audioEventsSubscription != null) {
            audioEventsSubscription.playbackFinished();
        }
    }
}
```

Le code suivant est un exemple de demande utilisant leAWS SDK for Java pour gérer les réponses du bot. Le code de cet exemple configure Amazon Lex V2 pour qu'il vous envoie une réponse audio.

```
package com.lex.streaming.sample;

import javazoom.jl.decoder.JavaLayerException;
import javazoom.jl.player.advanced.AdvancedPlayer;
import javazoom.jl.player.advanced.PlaybackEvent;
import javazoom.jl.player.advanced.PlaybackListener;
import software.amazon.awssdk.core.async.SdkPublisher;
import software.amazon.awssdk.services.lexruntimev2.model.AudioResponseEvent;
import software.amazon.awssdk.services.lexruntimev2.model.DialogActionType;
import software.amazon.awssdk.services.lexruntimev2.model.IntentResultEvent;
import software.amazon.awssdk.services.lexruntimev2.model.PlaybackInterruptionEvent;
import software.amazon.awssdk.services.lexruntimev2.model.StartConversationResponse;
import
 software.amazon.awssdk.services.lexruntimev2.model.StartConversationResponseEventStream;
import
 software.amazon.awssdk.services.lexruntimev2.model.StartConversationResponseHandler;
import software.amazon.awssdk.services.lexruntimev2.model.TextResponseEvent;
import software.amazon.awssdk.services.lexruntimev2.model.TranscriptEvent;

import java.io.IOException;
import java.io.UncheckedIOException;
import java.util.concurrent.CompletableFuture;

/**
 * The following class is responsible for processing events sent from the Amazon Lex
 bot. The bot sends multiple audio events,
 * so the following code concatenates those audio events and uses a publicly available
 Java audio player to play out the message to
```

```
 * the user.
 */
public class BotResponseHandler implements StartConversationResponseHandler {

    private final EventsPublisher eventsPublisher;

    private boolean lastBotResponsePlayedBack;
    private boolean isDialogStateClosed;
    private AudioResponse audioResponse;


    public BotResponseHandler(EventsPublisher eventsPublisher) {
        this.eventsPublisher = eventsPublisher;
        this.lastBotResponsePlayedBack = false;// At the start, we have not played back
 last response from bot.
        this.isDialogStateClosed = false; // At the start, the dialog state is open.
    }

    @Override
    public void responseReceived(StartConversationResponse startConversationResponse) {
        System.out.println("successfully established the connection with server.
 request id:" + startConversationResponse.responseMetadata().requestId()); // would
 have 2XX, request id.
    }

    @Override
    public void onEventStream(SdkPublisher<StartConversationResponseEventStream>
 sdkPublisher) {

        sdkPublisher.subscribe(event -> {
            if (event instanceof PlaybackInterruptionEvent) {
                handle((PlaybackInterruptionEvent) event);
            } else if (event instanceof TranscriptEvent) {
                handle((TranscriptEvent) event);
            } else if (event instanceof IntentResultEvent) {
                handle((IntentResultEvent) event);
            } else if (event instanceof TextResponseEvent) {
                handle((TextResponseEvent) event);
            } else if (event instanceof AudioResponseEvent) {
                handle((AudioResponseEvent) event);
            }
        });
    }
```

```
    @Override
    public void exceptionOccurred(Throwable throwable) {
        System.err.println("got an exception:" + throwable);
    }

    @Override
    public void complete() {
        System.out.println("on complete");
    }

    private void handle(PlaybackInterruptionEvent event) {
        System.out.println("Got a PlaybackInterruptionEvent: " + event);
    }

    private void handle(TranscriptEvent event) {
        System.out.println("Got a TranscriptEvent: " + event);
    }


    private void handle(IntentResultEvent event) {
        System.out.println("Got an IntentResultEvent: " + event);
        isDialogStateClosed =
DialogActionType.CLOSE.equals(event.sessionState().dialogAction().type());
    }

    private void handle(TextResponseEvent event) {
        System.out.println("Got an TextResponseEvent: " + event);
        event.messages().forEach(message -> {
            System.out.println("Message content type:" + message.contentType());
            System.out.println("Message content:" + message.content());
        });
    }

    private void handle(AudioResponseEvent event) {//Synthesize speech
        // System.out.println("Got a AudioResponseEvent: " + event);
        if (audioResponse == null) {
            audioResponse = new AudioResponse();
            //Start an audio player in a different thread.
            CompletableFuture.runAsync(() -> {
                try {
                    AdvancedPlayer audioPlayer = new AdvancedPlayer(audioResponse);

                    audioPlayer.setPlayBackListener(new PlaybackListener() {
                        @Override
```

```
                        public void playbackFinished(PlaybackEvent evt) {
                            super.playbackFinished(evt);

                            // Inform the Amazon Lex bot that the playback has
 finished.
                            eventsPublisher.playbackFinished();
                            if (isDialogStateClosed) {
                                lastBotResponsePlayedBack = true;
                            }
                        }
                    });
                    audioPlayer.play();
                } catch (JavaLayerException e) {
                    throw new RuntimeException("got an exception when using audio
 player", e);
                }
            });
        }

        if (event.audioChunk() != null) {
            audioResponse.write(event.audioChunk().asByteArray());
        } else {
            // The audio audio prompt has ended when the audio response has no
            // audio bytes.
            try {
                audioResponse.close();
                audioResponse = null;  // Prepare for the next audio prompt.
            } catch (IOException e) {
                throw new UncheckedIOException("got an exception when closing the audio
 response", e);
            }
        }
    }

    // The conversation with the Amazon Lex bot is complete when the bot marks the
 Dialog as DialogActionType.CLOSE
    // and any prompt playback is finished. For more information, see
    // https://docs.aws.amazon.com/lexv2/latest/dg/API_runtime_DialogAction.html.
    public boolean isConversationComplete() {
        return isDialogStateClosed && lastBotResponsePlayedBack;
    }

}
```

Pour configurer un robot afin qu'il réponde aux événements d'entrée par le biais de l'audio, vous devez d'abord vous abonner aux événements audio depuis Amazon Lex V2, puis configurer le bot pour qu'il fournisse une réponse audio aux événements d'entrée de l'utilisateur.

Le code suivant est unAWS SDK for Java exemple d'abonnement à des événements audio depuis Amazon Lex V2.

```
package com.lex.streaming.sample;

import org.reactivestreams.Subscriber;
import org.reactivestreams.Subscription;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.services.lexruntimev2.model.AudioInputEvent;
import software.amazon.awssdk.services.lexruntimev2.model.ConfigurationEvent;
import software.amazon.awssdk.services.lexruntimev2.model.DisconnectionEvent;
import software.amazon.awssdk.services.lexruntimev2.model.PlaybackCompletionEvent;
import
 software.amazon.awssdk.services.lexruntimev2.model.StartConversationRequestEventStream;

import javax.sound.sampled.AudioFormat;
import javax.sound.sampled.AudioInputStream;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.DataLine;
import javax.sound.sampled.LineUnavailableException;
import javax.sound.sampled.TargetDataLine;
import java.io.IOException;
import java.io.UncheckedIOException;
import java.nio.ByteBuffer;
import java.util.Arrays;
import java.util.concurrent.BlockingQueue;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.LinkedBlockingQueue;
import java.util.concurrent.atomic.AtomicLong;

public class AudioEventsSubscription implements Subscription {
    private static final AudioFormat MIC_FORMAT = new AudioFormat(8000, 16, 1, true,
 false);
    private static final String AUDIO_CONTENT_TYPE = "audio/lpcm; sample-rate=8000;
 sample-size-bits=16; channel-count=1; is-big-endian=false";
    //private static final String RESPONSE_TYPE = "audio/pcm; sample-rate=8000";
```

```
    private static final String RESPONSE_TYPE = "audio/mpeg";
    private static final int BYTES_IN_AUDIO_CHUNK = 320;
    private static final AtomicLong eventIdGenerator = new AtomicLong(0);

    private final AudioInputStream audioInputStream;
    private final Subscriber<? super StartConversationRequestEventStream> subscriber;
    private final EventWriter eventWriter;
    private CompletableFuture eventWriterFuture;


    public AudioEventsSubscription(Subscriber<? super
StartConversationRequestEventStream> subscriber) {
        this.audioInputStream = getMicStream();
        this.subscriber = subscriber;
        this.eventWriter = new EventWriter(subscriber, audioInputStream);
        configureConversation();
    }

    private AudioInputStream getMicStream() {
        try {
            DataLine.Info dataLineInfo = new DataLine.Info(TargetDataLine.class,
MIC_FORMAT);
            TargetDataLine targetDataLine = (TargetDataLine)
AudioSystem.getLine(dataLineInfo);

            targetDataLine.open(MIC_FORMAT);
            targetDataLine.start();

            return new AudioInputStream(targetDataLine);
        } catch (LineUnavailableException e) {
            throw new RuntimeException(e);
        }
    }

    @Override
    public void request(long demand) {
        // If a thread to write events has not been started, start it.
        if (eventWriterFuture == null) {
            eventWriterFuture = CompletableFuture.runAsync(eventWriter);
        }
        eventWriter.addDemand(demand);
    }

    @Override
```

```
    public void cancel() {
        subscriber.onError(new RuntimeException("stream was cancelled"));
        try {
            audioInputStream.close();
        } catch (IOException e) {
            throw new UncheckedIOException(e);
        }
    }

    public void configureConversation() {
        String eventId = "ConfigurationEvent-" +
String.valueOf(eventIdGenerator.incrementAndGet());

        ConfigurationEvent configurationEvent = StartConversationRequestEventStream
                .configurationEventBuilder()
                .eventId(eventId)
                .clientTimestampMillis(System.currentTimeMillis())
                .responseContentType(RESPONSE_TYPE)
                .build();

        System.out.println("writing config event");
        eventWriter.writeConfigurationEvent(configurationEvent);
    }

    public void disconnect() {

        String eventId = "DisconnectionEvent-" +
String.valueOf(eventIdGenerator.incrementAndGet());

        DisconnectionEvent disconnectionEvent = StartConversationRequestEventStream
                .disconnectionEventBuilder()
                .eventId(eventId)
                .clientTimestampMillis(System.currentTimeMillis())
                .build();

        eventWriter.writeDisconnectEvent(disconnectionEvent);

        try {
            audioInputStream.close();
        } catch (IOException e) {
            throw new UncheckedIOException(e);
        }

    }
```

```
    //Notify the subscriber that we've finished.
    public void stop() {
        subscriber.onComplete();
    }

    public void playbackFinished() {
        String eventId = "PlaybackCompletion-" +
String.valueOf(eventIdGenerator.incrementAndGet());

        PlaybackCompletionEvent playbackCompletionEvent =
StartConversationRequestEventStream
                .playbackCompletionEventBuilder()
                .eventId(eventId)
                .clientTimestampMillis(System.currentTimeMillis())
                .build();

        eventWriter.writePlaybackFinishedEvent(playbackCompletionEvent);
    }

    private static class EventWriter implements Runnable {
        private final BlockingQueue<StartConversationRequestEventStream> eventQueue;
        private final AudioInputStream audioInputStream;
        private final AtomicLong demand;
        private final Subscriber subscriber;

        private boolean conversationConfigured;

        public EventWriter(Subscriber subscriber, AudioInputStream audioInputStream) {
            this.eventQueue = new LinkedBlockingQueue<>();

            this.demand = new AtomicLong(0);
            this.subscriber = subscriber;
            this.audioInputStream = audioInputStream;
        }

        public void writeConfigurationEvent(ConfigurationEvent configurationEvent) {
            eventQueue.add(configurationEvent);
        }

        public void writeDisconnectEvent(DisconnectionEvent disconnectionEvent) {
            eventQueue.add(disconnectionEvent);
        }
```

```
        public void writePlaybackFinishedEvent(PlaybackCompletionEvent
playbackCompletionEvent) {
            eventQueue.add(playbackCompletionEvent);
        }

        void addDemand(long l) {
            this.demand.addAndGet(l);
        }

        @Override
        public void run() {
            try {

                while (true) {
                    long currentDemand = demand.get();

                    if (currentDemand > 0) {
                        // Try to read from queue of events.
                        // If nothing is in queue at this point, read the audio events
directly from audio stream.
                        for (long i = 0; i < currentDemand; i++) {

                            if (eventQueue.peek() != null) {
                                subscriber.onNext(eventQueue.take());
                                demand.decrementAndGet();
                            } else {
                                writeAudioEvent();
                            }
                        }
                    }
                }
            } catch (InterruptedException e) {
                throw new RuntimeException("interrupted when reading data to be sent to
server");
            } catch (Exception e) {
                e.printStackTrace();
            }
        }

        private void writeAudioEvent() {
            byte[] bytes = new byte[BYTES_IN_AUDIO_CHUNK];

            int numBytesRead = 0;
            try {
```

```
                numBytesRead = audioInputStream.read(bytes);
                if (numBytesRead != -1) {
                    byte[] byteArrayCopy = Arrays.copyOf(bytes, numBytesRead);

                    String eventId = "AudioEvent-" +
 String.valueOf(eventIdGenerator.incrementAndGet());

                    AudioInputEvent audioInputEvent =
 StartConversationRequestEventStream
                            .audioInputEventBuilder()

 .audioChunk(SdkBytes.fromByteBuffer(ByteBuffer.wrap(byteArrayCopy)))
                            .contentType(AUDIO_CONTENT_TYPE)
                            .clientTimestampMillis(System.currentTimeMillis())
                            .eventId(eventId).build();

                    //System.out.println("sending audio event:" + audioInputEvent);
                    subscriber.onNext(audioInputEvent);
                    demand.decrementAndGet();
                    //System.out.println("sent audio event:" + audioInputEvent);
                } else {
                    subscriber.onComplete();
                    System.out.println("audio stream has ended");
                }

            } catch (IOException e) {
                System.out.println("got an exception when reading from audio stream");
                System.err.println(e);
                subscriber.onError(e);
            }
        }
    }
}
```

L'AWS SDK for Javaexemple suivant configure le bot Amazon Lex V2 pour fournir une réponse audio aux événements d'entrée.

```
package com.lex.streaming.sample;

import java.io.IOException;
```

```
import java.io.InputStream;
import java.io.UncheckedIOException;
import java.util.Optional;
import java.util.concurrent.LinkedBlockingQueue;
import java.util.concurrent.TimeUnit;

public class AudioResponse extends InputStream{

    // Used to convert byte, which is signed in Java, to positive integer (unsigned)
    private static final int UNSIGNED_BYTE_MASK = 0xFF;
    private static final long POLL_INTERVAL_MS = 10;

    private final LinkedBlockingQueue<Integer> byteQueue = new LinkedBlockingQueue<>();

    private volatile boolean closed;

    @Override
    public int read() throws IOException {
        try {
            Optional<Integer> maybeInt;
            while (true) {
                maybeInt = Optional.ofNullable(this.byteQueue.poll(POLL_INTERVAL_MS,
 TimeUnit.MILLISECONDS));

                // If we get an integer from the queue, return it.
                if (maybeInt.isPresent()) {
                    return maybeInt.get();
                }

                // If the stream is closed and there is nothing queued up, return -1.
                if (this.closed) {
                    return -1;
                }
            }
        } catch (InterruptedException e) {
            throw new IOException(e);
        }
    }

    /**
     * Writes data into the stream to be offered on future read() calls.
     */
    public void write(byte[] byteArray) {
        // Don't write into the stream if it is already closed.
```

```
        if (this.closed) {
            throw new UncheckedIOException(new IOException("Stream already closed when
 attempting to write into it."));
        }

        for (byte b : byteArray) {
            this.byteQueue.add(b & UNSIGNED_BYTE_MASK);
        }
    }

    @Override
    public void close() throws IOException {
        this.closed = true;
        super.close();
    }
}
```

Codage du flux d'événements

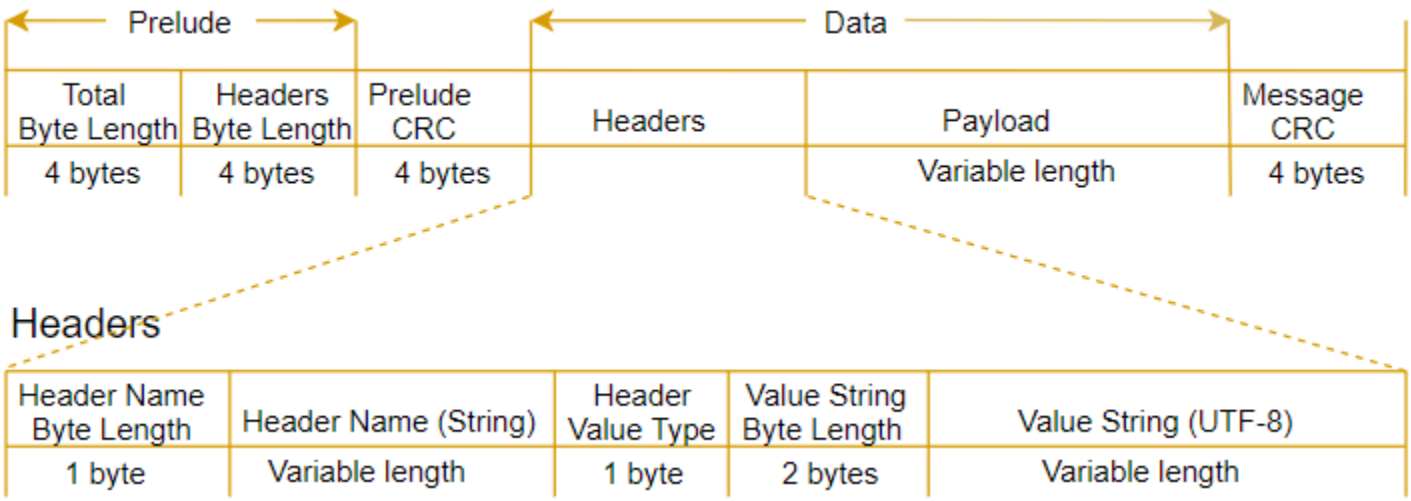
L'encodage de flux d'événements permet une communication bidirectionnelle par message entre un client et un serveur. Les trames de données envoyées au service de streaming Amazon Lex V2 sont codées dans ce format. La réponse d'Amazon Lex V2 utilise également ce codage.

Chaque message est constitué de deux sections : le préambule et les données. La section Prélude contient la longueur totale en octets du message et la longueur en octets combinée de tous les en-têtes. La section des données contient les en-têtes et une charge utile.

Chaque section se termine par un total de contrôle CRC entier à 4 octets de poids fort. Le message CRC checksum inclut la section prélude et la section données. Amazon Lex V2 utilise CRC32 (souvent appelé GZIP CRC32) pour calculer les deux CRC. Pour plus d'informations sur CRC32, consultez la rubrique Spécification du format de fichiers GZIP version 4.3.

La surcharge totale de message, incluant le préambule et les deux totaux de contrôle, s'élève à 16 octets.

Le schéma suivant montre les composants qui constituent un message et un en-tête. Il y a plusieurs en-têtes par message.



Chaque message comporte les composants suivants :

- Préambule : d'une taille toujours égale à 8 octets, il comprend deux champs de 4 octets.
  - Les 4 premiers octets : la longueur totale en octets. Il s'agit de l'entier de poids fort de l'ensemble du message, incluant le champ de 4 octets.
  - Second 4 octets : la longueur en octets des en-têtes. Il s'agit de l'entier de poids fort de la partie des en-têtes du message, sans le champ des en-têtes.
- CRC du préambule : le total de contrôle CRC de 4 octets de la partie préambule du message, sans le CRC. Le prélude possède un CRC distinct du CRC du message afin de garantir qu'Amazon Lex V2 puisse détecter immédiatement les informations de longueur d'octet corrompues sans provoquer d'erreurs telles que des dépassements de mémoire tampon.
- En-têtes : les métadonnées qui décrivent le message (type du message, de contenu, etc.). Les messages ont plusieurs en-têtes. Les en-têtes sont des paires clé-valeur dans lesquelles la clé est une chaîne UTF-8. Les en-têtes peuvent être insérés dans n'importe quel ordre dans la partie en-têtes du message et tout en-tête ne peut apparaître qu'une seule fois. Pour connaître les types d'en-tête requis, consultez les sections suivantes.
- Charge utile : contenu audio ou texte envoyé à Amazon Lex.
- CRC du message : le total de contrôle CRC de 4 octets du début du message au début du total de contrôle. Cela inclut tout ce qui se trouve dans le message à l'exception de la CRC elle-même.

Chaque en-tête comporte les composants suivants : Il y a plusieurs en-têtes par image.

- Longueur en octet du nom de l'en-tête : la longueur en octet du nom de l'en-tête.

- Nom de l'en-tête : le nom de l'en-tête, indiquant son type. Pour connaître les valeurs valides, consultez les descriptions d'image suivantes :
- Type de valeur d'en-tête : une énumération indiquant la valeur d'en-tête.
- Longueur en octet de la chaîne de valeur : la longueur en octet de la chaîne de valeur.
- Valeur de l'en-tête : la valeur de la chaîne de l'en-tête. Les valeurs valides de ce champ dépendent du type d'en-tête. Pour connaître les valeurs valides, consultez les descriptions d'image suivantes :

Permettre à votre bot d'être interrompu par votre utilisateur

Lorsque vous lancez un flux audio bidirectionnel entre un robot Amazon Lex V2 et votre application, vous pouvez configurer le bot pour qu'il écoute les entrées de l'utilisateur pendant qu'il renvoie une invite. L'utilisateur peut ainsi interrompre l'invite avant que le bot n'ait fini de la lire. Vous pouvez utiliser cette configuration lorsque l'utilisateur connaît peut-être déjà la réponse à une question, par exemple lorsqu'il est invité à fournir un code CVV.

Un robot sait quand l'utilisateur interrompt une invite lorsqu'il détecte une saisie utilisateur avant que votre application ne puisse renvoyer un`PlaybackCompletion` événement. Lorsque l'utilisateur interrompt un robot, celui-ci envoie un`PlaybackInterruptionEvent`.

Par défaut, l'utilisateur peut interrompre toute invite indiquant que le bot diffuse vers votre application. Vous pouvez modifier ce paramètre dans la console Amazon Lex V2.

Vous pouvez modifier la façon dont un utilisateur peut répondre à une invite en modifiant un créneau. Un slot fait partie d'une intention, et c'est le moyen par lequel l'utilisateur vous fournit les informations souhaitées. À chaque emplacement, l'utilisateur est invité à vous fournir ces informations. Pour en savoir plus sur les machines à sous, consultezComment ça marche.

Pour modifier si l'utilisateur peut interrompre une invite (console)

1. Connectez-vous à la console Amazon Lex V2AWS Management Console et ouvrez-la sur la console Amazon Lex V2.
2. Sous Bots, sélectionnez un robot.
3. Sous Langue, sélectionnez la langue du bot.
4. Choisissez Afficher les intentions.
5. Choisissez l'intention .
6. Pour Machines à sous, choisissez une machine à sous.

7. Sous Options avancées, choisissez Slot prompts.
8. Choisissez Plus d'options rapides.
9. Sélectionnez ou désélectionnez Les utilisateurs peuvent interrompre l'invite en cours de lecture.

Vous pouvez tester cette fonctionnalité en créant un robot doté de deux emplacements et en spécifiant que les utilisateurs ne peuvent pas interrompre une invite pour un seul emplacement. Si vous interrompez une invite interruptible, le bot envoie un événement d'interruption de lecture. Si vous interrompez un signal sans interruption, le message continue de jouer.

Permettre au bot d'attendre que l'utilisateur fournisse plus d'informations

Lorsque vous lancez un flux bidirectionnel depuis un bot Amazon Lex V2 vers votre application, vous pouvez configurer le bot pour qu'il attende que l'utilisateur fournisse des informations supplémentaires. Dans certains cas, un utilisateur peut ne pas être prêt à répondre à une demande. Par exemple, un utilisateur peut ne pas être prêt à fournir les informations de sa carte de crédit parce que son portefeuille se trouve dans une autre pièce.

En utilisant le comportement Wait and continue du bot Amazon Lex V2, les utilisateurs peuvent prononcer des phrases telles que « attendez une seconde » pour faire attendre au robot qu'ils trouvent les informations et les fournissent. Lorsque vous activez ce comportement, le bot envoie des rappels périodiques à l'utilisateur pour qu'il fournisse les informations. Il ne renvoie pas les événements de transcription car il n'y a aucun énoncé utilisateur à transcrire.

Le bot Amazon Lex V2 gère automatiquement une conversation en streaming. Vous n'avez pas besoin d'écrire de code supplémentaire pour activer cette fonctionnalité. Lorsqu'un robot est invité à attendre par l'utilisateur, le`state` «`Intent` est »`Waiting` et le`type` «`DialogAction` est`ElicitSlot` ». Vous pouvez utiliser ces informations pour personnaliser votre application en fonction de vos besoins. Par exemple, vous pouvez configurer votre application pour qu'elle diffuse de la musique lorsque l'utilisateur recherche sa carte de crédit.

Vous activez le comportement d'attente et de poursuite pour un créneau individuel. Pour en savoir plus sur les machines à sous, consultezComment ça marche.

Pour activer, attendez et continuez

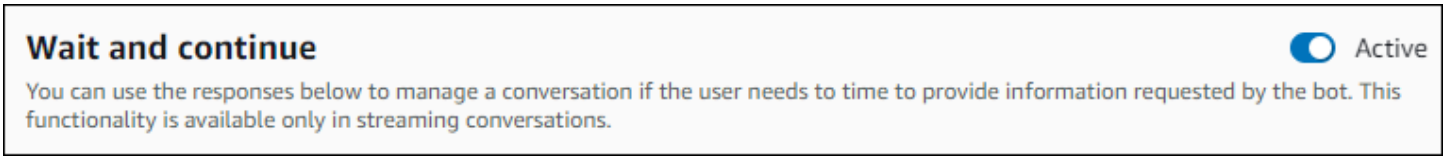
1. Connectez-vous à la console Amazon Lex V2AWS Management Console et ouvrez-la sur la console Amazon Lex V2.
2. Sous Bots, sélectionnez un robot.
3. Sous Langue, sélectionnez la langue du bot.
4. Choisissez Afficher les intentions.
5. Choisissez l'intention .
6. Sous Machines à sous, choisissez une machine à sous.
7. Sous Options avancées, choisissez Attendre et continuer.
8. Sous Attendre et continuer, spécifiez les champs suivants :

   - Réponse lorsque l'utilisateur souhaite que le robot attende — C'est ainsi que le bot répond lorsque l'utilisateur lui demande d'attendre les informations supplémentaires.
   - Réponse si l'utilisateur a besoin que le robot continue d'attendre : il s'agit de la réponse que le robot envoie pour rappeler à l'utilisateur qu'il attend toujours les informations. Vous pouvez modifier la fréquence à laquelle le bot rappelle à l'utilisateur.
   - Réponse lorsque l'utilisateur souhaite continuer : il s'agit de la réponse du bot lorsque l'utilisateur dispose des informations demandées.

Pour chaque réponse du bot, vous pouvez donner plusieurs variantes de la réponse, et l'une d'entre elles est présentée à l'utilisateur de manière aléatoire. Vous pouvez également choisir si ces réponses peuvent être interrompues par l'utilisateur.

Pour tester la fonctionnalité d'attente et de poursuite, configurez votre robot pour qu'il attende les entrées de l'utilisateur et lance un flux vers un robot Amazon Lex V2. Pour plus d'informations sur le streaming vers un bot, consultezUtilisation de l'API pour démarrer une conversation en streaming.

Il se peut que vous deviez désactiver l'attente et poursuivre les réponses. Utilisez le bouton Actif pour définir si les réponses attendre et continuer sont utilisées ou non.



Configuration des mises à jour de progression du traitement

Lorsque la fonction Lambda d'exécution d'une intention est appelée, le bot n'envoie pas de réponse tant que la fonction n'est pas terminée. Si l'exécution de la fonction Lambda prend plus de quelques secondes, l'utilisateur peut penser que le bot ne répond pas. Pour résoudre ce problème, vous pouvez configurer votre robot pour qu'il envoie des mises à jour à l'utilisateur pendant que la fonction Lambda de traitement est en cours d'exécution afin que l'utilisateur sache que le bot travaille toujours sur sa demande.

Lorsque vous ajoutez des mises à jour d'exécution à une intention, le robot répond au début de l'exécution et régulièrement pendant le traitement. Lorsque vous configurez la réponse de démarrage, vous pouvez spécifier un délai avant que le bot n'envoie la réponse. Cela vous permet de prendre en charge les cas où le traitement ne se termine pas assez rapidement. Lorsque vous configurez une réponse de mise à jour, vous spécifiez la fréquence à laquelle vous souhaitez que les mises à jour soient envoyées. Vous configurez également un délai d'expiration pour limiter la durée d'exécution de la fonction de traitement des commandes.

Vous pouvez également ajouter des réponses post-traitement à un robot. Cela permet au robot d'envoyer une réponse différente selon que l'exécution est réussie, échouée ou expirée.

Les mises à jour de traitement des commandes ne sont utilisées que lorsque vous interagissez avec un robot utilisant l'StartConversationopération. Vous pouvez utiliser la mise à jour après traitement lorsque vous interagissez avec le robot à l'aide RecognizeUtterancedes opérations StartConversationRecognizeText, et

Mises à jour relatives

Les mises à jour d'exécution sont envoyées alors que votre fonction Lambda répond à une intention. Lorsque vous activez les mises à jour du traitement, vous fournissez une réponse de début qui est envoyée au début du traitement et une réponse de mise à jour qui est envoyée périodiquement pendant le traitement.

Lorsque vous spécifiez une réponse de mise à jour, vous spécifiez également un délai d'expiration qui détermine la durée pendant laquelle la fonction de traitement peut s'exécuter. Vous pouvez spécifier un délai d'expiration pouvant aller jusqu'à 15 minutes (900 secondes).

Si vous désactivez les mises à jour d'exécution en les définissant`active` sur false dans la console ou en utilisant l'UpdateIntentopération CreateIntentor, le délai d'expiration spécifié pour les mises à jour d'exécution n'est pas utilisé et le délai d'attente par défaut de 30 secondes est utilisé à la place.

Si la fonction de traitement des commandes arrive à expiration, Amazon Lex V2 effectue l'une des trois opérations suivantes :

- La réponse post-traitement est configurée et active. Elle renvoie la réponse au délai d'expiration.
- La réponse post-traitement est configurée et n'est pas active. Elle renvoie une exception.
- La réponse post-traitement n'est pas configurée. Elle renvoie une exception.

Commencer la réponse

Amazon Lex V2 renvoie la réponse de démarrage lorsque la fonction de traitement Lambda est appelée au cours d'une conversation en streaming. Il indique généralement à l'utilisateur que la réalisation de l'intention prend un certain temps et qu'il doit attendre. La réponse de démarrage n'est pas renvoyée lorsque vous utilisez les`RecognizeUtterance` opérations`RecognizeText` ou.

Vous pouvez spécifier jusqu'à cinq messages de réponse. Amazon Lex V2 choisit l'un des messages à l'utilisateur.

Vous pouvez configurer un délai entre le moment où la fonction Lambda est appelée et le moment où la réponse de démarrage est renvoyée. La réponse de démarrage n'est pas renvoyée si la fonction Lambda termine son travail avant la fin du délai.

Vous pouvez utiliser le`active` bouton de la console ou de la FulfillmentUpdatesSpecificationstructure pour activer et désactiver la réponse de démarrage. Lorsque la valeur`active` est False, la réponse de démarrage n'est pas jouée.

Mettre à jour la réponse

Amazon Lex renvoie régulièrement la réponse de mise à jour lors d'une conversation en streaming alors que la fonction de traitement Lambda est en cours d'exécution. La réponse de mise à jour n'est pas lue lorsque vous utilisez les`RecognizeUtterance` opérations`RecognizeText` ou. Vous pouvez configurer la fréquence d'exécution de la réponse de mise à jour. Par exemple, vous pouvez afficher une réponse de mise à jour toutes les 30 secondes pendant que la fonction de traitement s'exécute pour informer l'utilisateur que le processus est en cours et qu'il doit continuer à attendre.

Vous pouvez spécifier jusqu'à cinq messages de mise à jour. Amazon Lex V2 choisit un message à diffuser à l'utilisateur. L'utilisation de plusieurs messages évite que les mises à jour soient répétitives.

Si l'utilisateur fournit une saisie vocale, DTMF ou texte alors que la fonction Lambda de traitement est en cours d'exécution, Amazon Lex V2 renvoie la réponse de mise à jour à l'utilisateur.

Si la fonction Lambda termine son travail avant la fin de la première période de mise à jour, la réponse de mise à jour n'est pas renvoyée.

Vous pouvez utiliser le`active` bouton de la console ou de la FulfillmentUpdatesSpecificationstructure pour activer et désactiver la réponse à la mise à jour. Lorsque la valeur`active` est false, la réponse de mise à jour n'est pas renvoyée.

Réponse après l'exécution

Amazon Lex V2 renvoie une réponse post-traitement lorsque la fonction de traitement se termine. Une réponse post-traitement peut être utilisée pour répondre à n'importe quelle intention, et pas seulement lors de la diffusion de conversations. La réponse après traitement permet à l'utilisateur de savoir que la fonction est terminée et qu'il en connaît le résultat.

Vous pouvez utiliser le`active` bouton de la console ou de la PostFulfillmentStatusSpecificationstructure pour activer et désactiver la réponse après traitement. Quand`active` c'est faux, la réponse n'est pas jouée.

Il existe trois types de réponses post-réalisation :

- Succès : renvoyé lorsque la fonction Lambda d'exécution termine son travail avec succès. Si les réponses post-traitement ne sont pas actives. Amazon Lex V2 exécute la prochaine action configurée.
- Délai d'expiration : renvoyé si la fonction Lambda ne termine pas son travail avant la fin du délai d'expiration configuré. Si les réponses après traitement ne sont pas actives, Amazon Lex V2 renvoie une exception.
- Échec : renvoyé lorsque la fonction Lambda renvoie l'état indiqué`Failed` dans la réponse ou lorsqu'Amazon Lex V2 rencontre une erreur lors de la réalisation de l'objectif. Si les réponses après traitement ne sont pas actives, Amazon Lex V2 renvoie une exception.

Vous pouvez spécifier jusqu'à cinq messages pour chaque type. Amazon Lex V2 choisit l'un des messages à l'utilisateur.

Contrairement aux réponses en réalisation et celles sans streaming, les réponses post-réalisation sont lues aussi bien pour les conversations en streaming que pour celles sans streaming.

Vous avez également la possibilité de remplacer ces messages en configurant la fonction Lambda pour qu'elle renvoie un message après traitement.

> **Note**
>
> Si l'intention comporte une réponse finale, elle est renvoyée après la réponse post-traitement.

Exemple de post-exécution

Pour mieux comprendre la réponse post-traitement, prenons, à titre d'exemple, un *BookTrip*bot créé pour aider à planifier un voyage, avec une *BookFlight*intention, configuré avec une fonction Lambda de traitement qui réserve le vol du client auprès d'une compagnie aérienne. Une fois les créneaux pour *BookFlight*obtenus, Amazon Lex V2 invoque la fonction Lambda de traitement des commandes. Au cours de ce processus d'réalisation, l'un des trois résultats suivants peut se produire :

- Succès — Le vol a été réservé avec succès.
- Délai d'expiration : le processus de réservation prend plus de temps que le délai d'exécution Lambda configuré (par exemple, si la compagnie aérienne ne peut pas être contactée dans le délai imparti).
- Échec — La réservation échoue pour une autre raison.

Vous pouvez tirer parti de la réponse post-traitement pour fournir une réponse plus significative à vos clients dans chacune de ces situations. Des exemples pour chaque situation sont les suivants :

- Réponse réussie — « Nous avons réussi à réserver votre billet et nous vous avons envoyé un e-mail de confirmation. N'hésitez pas à nous contacter en utilisant les coordonnées fournies dans cet e-mail si vous avez des questions. »
- Délai de réponse : « En raison du trafic intense sur nos systèmes, la réservation de votre billet prend plus de temps que prévu. Votre demande est dans notre file d'attente et nous vous avons envoyé un e-mail avec le numéro de référence correspondant à cette demande. Une fois que nous aurons réservé le billet, nous vous enverrons une confirmation de réservation. N'hésitez pas à nous contacter en utilisant les coordonnées fournies dans cet e-mail si vous avez des questions. »

> **Note**
>
> Si vous ne configurez pas de message de temporisation, Lex génère une erreur 4XX correspondant au cas d'utilisation.

- Réponse en cas d'échec — « Malheureusement, nous n'avons pas pu réserver votre billet. Nous avons envoyé un e-mail contenant des informations concernant le problème rencontré lors de la réservation. »

Configuration des délais pour la capture des données saisies par l'utilisateur

L'API de streaming Amazon Lex V2 permet à un robot de détecter automatiquement les énoncés saisis par l'utilisateur. Lorsque vous créez une intention ou un créneau, vous pouvez configurer certains aspects d'un énoncé, tels que la durée maximale d'un énoncé, le délai d'attente pendant la saisie par l'utilisateur ou le caractère final de la saisie DTMF. Vous pouvez personnaliser le comportement d'un robot en fonction de votre cas d'utilisation. Par exemple, vous pouvez limiter le nombre de chiffres d'un numéro de carte de crédit à 16.

Vous pouvez également configurer des délais d'expiration à l'aide d'attributs de session lorsque vous démarrez une conversation avec un bot, et les remplacer dans votre fonction Lambda si nécessaire.

Les clés de configuration d'un attribut utilisent la syntaxe suivante :

```
x-amz-lex:<InputType>:<BehaviorName>:<IntentName>:<SlotName>
```

`InputType` peut être **`audio`**, **`dtmf`** ou **`text`**.

Vous pouvez configurer les paramètres par défaut pour toutes les intentions ou tous les emplacements d'un robot en les spécifiant***** comme intention ou nom d'emplacement. Tous les paramètres spécifiques à l'intention ou à l'emplacement ont priorité sur les paramètres par défaut.

Amazon Lex V2 fournit des attributs de session prédéfinis pour gérer le fonctionnement StartConversationdes opérations avec du texte, de la voix ou des entrées DTMF destinées à votre bot. Tous les attributs prédéfinis sont dans l'espace de noms `x-amz-lex`.

Vous pouvez configurer les paramètres par défaut pour toutes les intentions, emplacements ou sous-emplacements d'un robot en les spécifiant* comme intention ou nom d'emplacement. Tous les paramètres spécifiques à l'intention ou à l'emplacement ont priorité sur les paramètres par défaut. Utilisez ces modèles pour tous les délais d'attente ci-dessous.

Pour le sous-emplacement d'un emplacement composite, vous pouvez le séparer par`.`. Par exemple :

```
<slotName>.<subSlotName>
```

```
x-amz-lex:allow-interrupt:<intentName>:<slotName>.<subSlotName>
```

Expression	Scénario
Intention : Slot. SubSlot	Applicable uniquement au sous-slot nommé «SubSlot » à l'intérieur du slot composite nommé « Slot »
Intention : Slot. *	Applicable à tout sous-emplacement à l'intérieur d'un emplacement composite nommé « Slot »
Intention : *. SubSlot	Applicable uniquement au sous-emplacement nommé «SubSlot » à l'intérieur d'un emplacement composite
Intention : * . *	Applicable à n'importe quel sous-emplacement à l'intérieur de n'importe quel emplacement composite

Comportement d'interruption

Vous pouvez configurer le comportement d'interruption du bot. L'attribut est défini dans Amazon Lex V2.

Autoriser l'interruption

```
x-amz-lex:allow-interrupt:<intentName>:<slotName>
```

Définit si l'utilisateur peut interrompre l'invite lue par le bot Amazon Lex V2. Vous pouvez le désactiver de manière sélective.

Valeur par défaut : VRAI

Délais de saisie vocale

Vous pouvez définir des valeurs de délai pour les interactions vocales avec votre bot à l'aide des attributs de session. Les attributs sont définis dans Amazon Lex V2. Ces attributs vous permettent de spécifier le temps pendant lequel Amazon Lex V2 attend qu'un client ait fini de parler avant de recueillir le discours d'entrée.

Tous ces attributs se trouvent dans l'espace `dex-amz-lex:audio` noms.

Longueur maximale de l'énoncé

```
x-amz-lex:audio:max-length-ms:<intentName>:<slotName>
```

Définit la durée pendant laquelle Amazon Lex V2 attend avant que la saisie vocale ne soit tronquée et que la voix soit renvoyée à votre application. Vous pouvez augmenter la longueur de la saisie lorsque vous attendez des réponses longues ou si vous souhaitez donner plus de temps aux clients pour fournir des informations.

Par défaut : 13 000 millisecondes (13 secondes). La valeur maximale est de 15 000 millisecondes (15 secondes)

Si vous définissez l'`max-length-ms`attribut sur plus de 15 000 millisecondes, la valeur par défaut sera de 15 000 millisecondes.

délai d'attente de connexion vocale

```
x-amz-lex:audio:start-timeout-ms:<intentName>:<slotName>
```

Combien de temps un robot attend avant de supposer que le client ne va pas parler. Vous pouvez augmenter le temps dans les situations où le client peut avoir besoin de plus de temps pour trouver ou se souvenir d'informations avant de parler. Par exemple, vous pouvez donner aux clients le temps de sortir leur carte de crédit afin qu'ils puissent saisir le numéro.

Par défaut : 4 000 millisecondes (4 secondes)

délai d'attente du délai d'attente

```
x-amz-lex:audio:end-timeout-ms:<intentName>:<slotName>
```

Combien de temps un robot attend une fois que le client arrête de parler pour supposer que l'énoncé est terminé. Vous pouvez augmenter la durée dans les situations où des périodes de silence sont attendues lors de la saisie.

Par défaut : 600 millisecondes (0,6 seconde)

Autoriser l'entrée audio

```
x-amz-lex:allow-audio-input:<intentName>:<slotName>
```

Vous pouvez activer cet attribut afin que le bot accepte les entrées de l'utilisateur uniquement via la modalité audio. Le bot n'acceptera pas d'entrée audio si cet indicateur est défini sur false. La valeur indique true.

Valeur par défaut : VRAI

Délais de saisie de texte

Utilisez l'attribut de session suivant pour spécifier le comportement de votre bot en mode conversation textuelle.

Cet attribut se trouve dans l'espace `dex-amz-lex:text` noms.

délai d'délai d'délai d'attente du délai

```
x-amz-lex:text:start-timeout-ms:<intentName>:<slotName>
```

Combien de temps le bot attend avant de demander à nouveau à un client de saisir du texte. Vous pouvez augmenter le délai dans les situations où vous souhaitez laisser au client plus de temps pour rechercher ou se souvenir des informations avant de saisir du texte. Par exemple, vous pouvez souhaiter intégrer le délai d'attente des informations de commande. Vous pouvez également réduire le seuil pour inviter les clients plus tôt.

défaut : 30 000 millisecondes 30 secondes (30 secondes)

Configuration pour l'entrée DTMF

Utilisez les attributs de session suivants pour spécifier la manière dont votre robot Amazon Lex V2 répond aux entrées DTMF lors d'une conversation audio.

Tous ces attributs se trouvent dans l'espace `dex-amz-lex:dtmf` noms.

Caractère de suppression

```
x-amz-lex:dtmf:deletion-character:<intentName>:<slotName>
```

Le caractère DTMF qui efface les chiffres DTMF accumulés et met immédiatement fin à la saisie.

Par défaut : *

Personnage final

```
x-amz-lex:dtmf:end-character:<intentName>:<slotName>
```

Le caractère DTMF qui met immédiatement fin à la saisie. Si l'utilisateur n'appuie pas sur ce caractère, la saisie se termine après le délai imparti.

Par défaut : #

délai d'attente de délai d'attente

```
x-amz-lex:dtmf:end-timeout-ms:<intentName>:<slotName>
```

Combien de temps le robot doit attendre à partir de la dernière saisie de caractères DTMF avant de supposer que la saisie est terminée.

Par défaut : 5 000 millisecondes (5 secondes)

Nombre maximum de chiffres DTMF par énoncé

```
x-amz-lex:dtmf:max-length:<intentName>:<slotName>
```

Nombre maximum de chiffres DTMF autorisés dans un énoncé. Par exemple, vous pouvez définir cette valeur sur 16 pour limiter le nombre de caractères pouvant être saisis pour un numéro de carte de crédit. Cette valeur ne peut pas être augmentée.

Par défaut : 1024 caractères

Autoriser la saisie DTMF

Vous pouvez définir le type d'entrée que le bot peut accepter à l'aide des attributs de session. Les attributs sont définis dans Amazon Lex V2.

```
x-amz-lex:allow-dtmf-input:<intentName>:<slotName>
```

Vous pouvez activer cet attribut afin que le bot accepte les entrées de l'utilisateur via la modalité DTMF. Le bot n'accepte pas l'entrée DTMF si cet indicateur est défini sur false. La valeur indique true.

Valeur par défaut : VRAI

Importation et exportation

Vous pouvez exporter la définition d'un robot, ses paramètres régionaux ou un vocabulaire personnalisé, puis le réimporter pour créer une nouvelle ressource ou remplacer une ressource existante dans un AWS compte. Par exemple, vous pouvez exporter un bot depuis un compte de test, puis créer une copie du bot dans votre compte de production. Vous pouvez également copier un bot d'une AWS région vers une autre région.

Vous pouvez modifier les ressources de la ressource exportée avant de l'importer. Par exemple, vous pouvez exporter un bot puis modifier le fichier JSON d'un emplacement afin d'ajouter ou de supprimer des énoncés de sollicitation de valeurs d'emplacement dans un emplacement spécifique. Une fois que vous avez terminé de modifier la définition, vous pouvez importer le fichier modifié.

Rubriques

Exporting

Vous exportez un bot, ses paramètres régionaux ou un vocabulaire personnalisé à l'aide de la console ou de l'`CreatExport`opération. Vous spécifiez la ressource à exporter et vous pouvez fournir un mot de passe facultatif pour protéger le fichier .zip lorsque vous démarrez une exportation. Après avoir téléchargé le fichier .zip, vous devez utiliser le mot de passe pour accéder au fichier avant de pouvoir l'utiliser. Pour plus d'informations, veuillez consulter Utilisation d'un mot de passe lors de l'importation ou de l'exportation.

L'exportation est une opération asynchrone. Une fois que vous avez lancé l'exportation, vous pouvez utiliser la console ou l'`DescribeExport`opération pour suivre la progression de l'exportation. Une fois l'exportation terminée, la console ou l'`DescribeExport`opération affiche le statut et la console télécharge le fichier .zip d'exportation vers votre navigateur. `COMPLETED` Si vous utilisez cette `DescribeExport` opération, Amazon Lex V2 fournit une URL Amazon S3 pré-signée où vous pouvez télécharger les résultats de l'exportation. L'URL de téléchargement n'est disponible que pendant cinq minutes, mais vous pouvez obtenir une nouvelle URL en relançant l'`DescribeExport`opération.

Vous pouvez consulter l'historique des exportations d'une ressource à l'aide de la console ou de l'`ListExports`opération. Les résultats indiquent les exportations ainsi que leur état actuel. Une exportation est disponible dans l'historique pendant sept jours.

Lorsque vous exportez la `Draft` version d'un bot ou de ses paramètres régionaux, il est possible que la définition du fichier JSON soit dans un état incohérent car la `Draft` version d'un bot ou de ses paramètres régionaux peut être modifiée pendant l'exportation. Si la `Draft` version est modifiée pendant son exportation, les modifications peuvent ne pas être incluses dans le fichier d'exportation.

Lorsque vous exportez les paramètres régionaux d'un bot, Amazon Lex exporte toutes les informations qui définissent les paramètres régionaux, notamment les paramètres régionaux, le vocabulaire personnalisé, les intentions, les types d'emplacements et les emplacements.

Lorsque vous exportez un bot, Amazon Lex exporte tous les paramètres régionaux définis pour le bot, y compris les intentions, les types d'emplacements et les emplacements. Les éléments suivants ne sont pas exportés avec un bot :

- Alias de robot
- Rôle ARN associé à un bot
- Tags associés aux robots et à leurs alias
- Hooks de code Lambda associés à un alias de bot

L'ARN du rôle et les balises sont saisis en tant que paramètres de demande lorsque vous importez un bot. Vous devez créer des alias de bot et attribuer des crochets de code Lambda après l'importation, si nécessaire.

Vous pouvez supprimer une exportation et le fichier .zip associé à l'aide de la console ou de l'`DeleteExport`opération.

Pour un exemple d'exportation d'un bot à l'aide de la console, consultezExporter un bot (console).

Autorisations IAM requises pour exporter

Pour exporter des robots, des paramètres régionaux de robots et des vocabulaires personnalisés, l'utilisateur qui exécute l'exportation doit disposer des autorisations IAM suivantes.

API	• Actions IAM requises	Ressource
CreateExport	• CreateExport	Bot

API	• Actions IAM requises	Ressource
UpdateExport	• UpdateExport	Bot
DescribeExport	• DescribeExport • DescribeBot • DescribeCustomVocabulary • DescribeLocale • DescribeIntent • DescribeSlot • DescribeSlotType • ListLocale • ListIntent • ListSlot • ListSlotType	Bot
DescribeExportpour des vocabulaires personnalisés	• DescribeExport • DescribeCustomVocabulary	bot
DeleteExport	• DeleteExport	Bot
ListExports	• ListExports	*

Pour obtenir un exemple de politique IAM, consultez Autoriser un utilisateur à exporter des robots et des paramètres régionaux de robots .

Exporter un bot (console)

Vous pouvez exporter un robot à partir de la liste des robots, de la liste des versions ou de la page de détails des versions. Lorsque vous choisissez une version, Amazon Lex V2 exporte cette version. Les instructions suivantes supposent que vous commencez à exporter le bot à partir de la liste des robots, mais lorsque vous commencez avec une version, les étapes sont les mêmes.

Pour exporter un bot à l'aide de la console

1. Connectez-vous à la console Amazon Lex V2 AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/lexv2/home.
2. Dans la liste des robots, sélectionnez le bot à exporter.
3. Dans Action, choisissez Exporter.
4. Choisissez la version du bot, la plateforme et le format d'exportation.
5. (Facultatif) Entrez un mot de passe pour le fichier .zip. La fourniture d'un mot de passe permet de protéger l'archive de sortie.
6. Choisissez Export (Exporter).

Après avoir lancé l'exportation, vous revenez à la liste des robots. Pour suivre la progression de l'exportation, utilisez la liste de l'historique des importations/exportations. Lorsque l'état de l'exportation est terminé, la console télécharge automatiquement le fichier .zip sur votre ordinateur.

Pour télécharger à nouveau l'exportation, dans la liste d'importation/exportation, choisissez l'exportation, puis cliquez sur Télécharger. Vous pouvez fournir un mot de passe pour le fichier .zip téléchargé.

Pour exporter le langage d'un bot

1. Connectez-vous à la console Amazon Lex V2 AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/lexv2/home.
2. Dans la liste des robots, choisissez celui dont vous souhaitez exporter la langue.
3. Dans Ajouter des langues, choisissez Afficher les langues.
4. Dans la liste Toutes les langues, choisissez la langue à exporter.
5. Dans Action, choisissez Exporter.
6. Choisissez la version, la plateforme et le format du bot.
7. (Facultatif) Entrez un mot de passe pour le fichier .zip. La fourniture d'un mot de passe permet de protéger l'archive de sortie.
8. Choisissez Export (Exporter).

Après avoir lancé l'exportation, vous revenez à la liste des langues. Pour suivre la progression de l'exportation, utilisez la liste de l'historique des importations/exportations. Lorsque l'état de l'exportation est terminé, la console télécharge automatiquement le fichier .zip sur votre ordinateur.

Pour télécharger à nouveau l'exportation, dans la liste d'importation/exportation, choisissez l'exportation, puis cliquez sur Télécharger. Vous pouvez fournir un mot de passe pour le fichier .zip téléchargé.

Importation

Pour utiliser la console pour importer un robot précédemment exporté, ses paramètres régionaux ou son vocabulaire personnalisé, vous devez indiquer l'emplacement du fichier sur votre ordinateur local et le mot de passe facultatif permettant de déverrouiller le fichier. Pour voir un exemple, consultez Importer un bot (console).

Lorsque vous utilisez l'API, l'importation d'une ressource se fait en trois étapes :

1. Créez une URL de téléchargement à l'aide de l'`CreateUploadUrl`opération. Il n'est pas nécessaire de créer une URL de téléchargement lorsque vous utilisez la console.
2. Chargez le fichier .zip qui contient la définition de la ressource.
3. Lancez l'importation avec l'`StartImport`opération.

L'URL de téléchargement est une URL Amazon S3 pré-signée avec autorisation d'écriture. L'URL est disponible pendant cinq minutes après sa génération. Si vous protégez le fichier .zip par mot de passe, vous devez fournir le mot de passe lorsque vous démarrez l'importation. Pour plus d'informations, veuillez consulter Utilisation d'un mot de passe lors de l'importation ou de l'exportation.

Une importation est un processus asynchrone. Vous pouvez suivre la progression d'une importation à l'aide de la console ou de l'`DescribeImport`opération.

Lorsque vous importez un bot ou ses paramètres régionaux, il peut y avoir des conflits entre les noms de ressources du fichier d'importation et les noms de ressources existants dans Amazon Lex V2. Amazon Lex V2 peut gérer le conflit de trois manières :

- Échec en cas de conflit : l'importation s'arrête et aucune ressource n'est importée depuis le fichier .zip d'importation.
- Remplacer : Amazon Lex V2 importe toutes les ressources depuis le fichier .zip d'importation et remplace toute ressource existante par la définition du fichier d'importation.
- Ajouter : Amazon Lex V2 importe toutes les ressources depuis le fichier .zip d'importation et les ajoute à toute ressource existante avec la définition du fichier d'importation. Ceci n'est disponible que pour les paramètres régionaux du bot.

Vous pouvez consulter la liste des importations vers une ressource à l'aide de la console ou de l'`ListImports`opération. Les importations restent dans la liste pendant sept jours. Vous pouvez utiliser la console ou l'`DescribeImport`opération pour voir les détails d'une importation spécifique.

Vous pouvez également supprimer une importation et le fichier .zip associé à l'aide de la console ou de l'`DeleteImport`opération.

Pour un exemple d'importation d'un bot à l'aide de la console, consultezImporter un bot (console).

Autorisations IAM requises pour l'importation

Pour importer des robots, des paramètres régionaux de robots et des vocabulaires personnalisés, l'utilisateur qui exécute l'importation doit disposer des autorisations IAM suivantes.

API	Actions IAM requises	Ressource
CreateUploadUrl	• CreateUploadUrl	*
StartImportpour les robots et leurs paramètres régionaux	• StartImport • je suis : PassRole • CreateBot • CreateCustomVocabulary • CreateLocale • CreateIntent • CreateSlot • CreateSlotType • UpdateBot • UpdateCustomVocabulary • UpdateLocale • UpdateIntent • UpdateSlot • UpdateSlotType • DeleteBot • DeleteCustomVocabulary • DeleteLocale	1. Pour importer un nouveau bot : bot, alias de bot. 2. Pour remplacer un bot existant : bot. 3. Pour importer un nouveau paramètre régional : bot.

API	Actions IAM requises	Ressource
	• DeleteIntent • DeleteSlot • DeleteSlotType	
StartImportpour des vocabulaires personnalisés	• StartImport • CreateCustomVocabulary • DeleteCustomVocabulary • UpdateCustomVocabulary	bot
DescribeImport	• DescribeImport	Bot
DeleteImport	• DeleteImport	Bot
ListImports	• ListImports	*

Pour obtenir un exemple de politique IAM, consultez Autoriser un utilisateur à importer des robots et des paramètres régionaux de robots .

Importer un bot (console)

Pour importer un bot à l'aide de la console

1. Connectez-vous à la console Amazon Lex V2 AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/lexv2/home.
2. Dans Action, choisissez Importer.
3. Dans le fichier d'entrée, donnez un nom au robot, puis choisissez le fichier .zip qui contient les fichiers JSON qui définissent le robot.
4. Si le fichier .zip est protégé par mot de passe, entrez le mot de passe du fichier .zip. La protection de l'archive par mot de passe est facultative, mais elle permet de protéger le contenu.
5. Créez ou saisissez le rôle IAM qui définit les autorisations pour votre bot.
6. Indiquez si votre bot est soumis à la Loi sur la protection de la vie privée des enfants en ligne (COPPA).

7. Définissez un paramètre de délai d'inactivité pour votre bot. Si vous ne fournissez pas de valeur, la valeur du fichier zip est utilisée. Si le fichier .zip ne contient aucun paramètre de délai d'expiration, Amazon Lex V2 utilise la valeur par défaut de 300 secondes (cinq minutes).
8. (Facultatif) Ajoutez des balises pour votre bot.
9. Choisissez si vous souhaitez avertir en cas de remplacement de robots existants portant le même nom. Si vous activez les avertissements, si le bot que vous importez doit remplacer un bot existant, vous recevez un avertissement et le bot n'est pas importé. Si vous désactivez les avertissements, le bot importé remplace le bot existant portant le même nom.
10. Choisissez Import (Importer).

Après avoir lancé l'importation, vous revenez à la liste des robots. Pour suivre la progression de l'importation, utilisez la liste de l'historique des importations/exportations. Lorsque le statut de l'importation est terminé, vous pouvez choisir le bot dans la liste des robots pour le modifier ou le créer.

Pour importer la langue d'un bot

1. Connectez-vous à la console Amazon Lex V2 AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/lexv2/home.
2. Dans la liste des robots, choisissez le bot vers lequel vous souhaitez importer une langue.
3. Dans Ajouter des langues, choisissez Afficher les langues.
4. Dans Action, choisissez Importer.
5. Dans Fichier d'entrée, choisissez le fichier contenant la langue à importer. Si vous avez protégé le fichier .zip, saisissez le mot de passe dans Mot de passe.
6. Dans Langue, choisissez la langue sous laquelle vous souhaitez importer. La langue ne doit pas nécessairement correspondre à celle du fichier d'importation. Vous pouvez copier les intentions d'une langue à l'autre.
7. Dans Voice, choisissez la voix d'Amazon Polly à utiliser pour l'interaction vocale, ou choisissez Aucune pour un robot utilisant uniquement du texte.
8. Dans Seuil du score de confiance, entrez le seuil auquel Amazon Lex V2 insère le `AMAZON.FallbackIntentAMAZON.KendraSearchIntent`, le ou les deux lors du renvoi d'intentions alternatives.
9. Choisissez si vous souhaitez avertir en cas de remplacement d'une langue existante. Si vous activez les avertissements, si la langue que vous importez est susceptible de remplacer une

langue existante, vous recevez un avertissement et la langue n'est pas importée. Si vous désactivez les avertissements, la langue importée remplace la langue existante.

10. Choisissez Importer pour commencer à importer la langue.

Après avoir lancé l'importation, vous revenez à la liste des langues. Pour suivre la progression de l'importation, utilisez la liste de l'historique des importations/exportations. Lorsque l'état de l'importation est terminé, vous pouvez choisir la langue dans la liste des robots pour modifier ou créer le bot.

Utilisation d'un mot de passe lors de l'importation ou de l'exportation

Amazon Lex V2 peut protéger par mot de passe vos archives d'exportation ou lire vos archives d'importation protégées à l'aide de la compression de fichiers .zip standard. Vous devez toujours protéger par mot de passe vos archives d'importation et d'exportation.

Amazon Lex V2 envoie votre archive d'exportation vers un compartiment S3, et vous pouvez y accéder via une URL S3 pré-signée. L'URL n'est disponible que pendant cinq minutes. L'archive est accessible à toute personne ayant accès à l'URL de téléchargement. Pour protéger les données de l'archive, saisissez un mot de passe lorsque vous exportez la ressource. Si vous devez récupérer l'archive après l'expiration de l'URL, vous pouvez utiliser la console ou l'`DescribeExport`opération pour obtenir une nouvelle URL.

Si vous perdez le mot de passe d'une archive d'exportation, vous pouvez créer un nouveau mot de passe pour un fichier existant en choisissant Télécharger dans le tableau de l'historique des importations/exportations ou en utilisant l'`UpdateExport`opération. Si vous choisissez Télécharger dans le tableau d'historique pour une exportation et que vous ne fournissez pas de mot de passe, Amazon Lex V2 télécharge un fichier zip non protégé.

Format JSON pour l'importation et l'exportation

Vous importez et exportez des robots, des paramètres régionaux de robots ou des vocabulaires personnalisés depuis Amazon Lex V2 à l'aide d'un fichier .zip contenant des structures JSON décrivant les parties de la ressource. Lorsque vous exportez une ressource, Amazon Lex V2 crée le fichier .zip et le met à votre disposition à l'aide d'une URL présignée Amazon S3. Lorsque vous importez une ressource, vous devez créer un fichier .zip contenant les structures JSON et le télécharger vers une URL pré-signée S3.

Amazon Lex crée la structure de répertoires suivante dans le fichier .zip lorsque vous exportez un bot. Lorsque vous exportez les paramètres régionaux d'un bot, seule la structure située sous les paramètres régionaux est exportée. Lorsque vous exportez un vocabulaire personnalisé, seule la structure du vocabulaire personnalisé est exportée.

```
BotName_BotVersion_ExportID_LexJson.zip
            -or-
BotName_BotVersion_LocaleId_ExportId_LEX_JSON.zip
        --> manifest.json
        --> BotName
        ----> Bot.json
        ----> BotLocales
        ------> Locale_A
        --------> BotLocale.json
        --------> Intents
        ----------> Intent_A
        ------------> Intent.json
        ------------> Slots
        --------------> Slot_A
        ----------------> Slot.json
        --------------> Slot_B
        ----------------> Slot.json
        ----------> Intent_B
                       ...
        --------> SlotTypes
        ----------> SlotType_A
        ------------> SlotType.json
        ----------> SlotType_B
                        ...
        --------> CustomVocabulary
        ------------> CustomVocabulary.json

        ------> Locale_B
                        ...
```

Structure du fichier manifeste

Le fichier manifeste contient les métadonnées du fichier d'exportation.

```
{
    "metadata": {
```

```
        "schemaVersion": "1.0",
        "fileFormat": "LexJson",
        "resourceType": "Bot | BotLocale | CustomVocabulary"
    }
}
```

Structure des fichiers du bot

Le fichier du bot contient les informations de configuration du bot.

```
{
    "name": "BotName",
    "identifier": "identifier",
    "version": "number",
    "description": "description",
    "dataPrivacy": {
        "childDirected": true | false
    },
    "idleSessionTTLInSeconds": seconds
}
```

Structure du fichier de paramètres régionaux du bot

Le fichier de paramètres régionaux du bot contient une description des paramètres régionaux ou de la langue d'un bot. Lorsque vous exportez un bot, le fichier .zip peut contenir plusieurs fichiers de paramètres régionaux du bot. Lorsque vous exportez les paramètres régionaux d'un bot, le fichier zip ne contient qu'un seul paramètre régional.

```
{
    "name": "locale name",
    "identifier": "locale ID",
    "version": "number",
    "description": "description",
    "voiceSettings": {
        "voiceId": "voice",
        "engine": "standard | neural
    },
    "nluConfidenceThreshold": number
}
```

Structure du fichier d'intention

Le fichier d'intention contient les informations de configuration d'une intention. Le fichier .zip contient un fichier d'intention pour chaque intention dans un environnement régional spécifique.

Voici un exemple de structure JSON pour l'BookCarintention de l'exemple de BookTrip bot. Pour un exemple complet de la structure JSON d'une intention, consultez l'CreateIntentopération.

```
{
    "name": "BookCar",
    "identifier": "891RWHHICO",
    "description": "Intent to book a car.",
    "parentIntentSignature": null,
    "sampleUtterances": [
        {
            "utterance": "Book a car"
        },
        {
            "utterance": "Reserve a car"
        },
        {
            "utterance": "Make a car reservation"
        }
    ],
    "intentConfirmationSetting": {
        "confirmationPrompt": {
            "messageGroupList": [
                {
                    "message": {
                        "plainTextMessage": {
                            "value": "OK, I have you down for a {CarType} hire in
 {PickUpCity} from {PickUpDate} to {ReturnDate}.  Should I book the reservation?"
                        },
                        "ssmlMessage": null,
                        "customPayload": null,
                        "imageResponseCard": null
                    },
                    "variations": null
                }
            ],
            "maxRetries": 2
        },
        "declinationResponse": {
```

```
            "messageGroupList": [
                {
                    "message": {
                        "plainTextMessage": {
                            "value": "OK, I have cancelled your reservation in
progress."
                        },
                        "ssmlMessage": null,
                        "customPayload": null,
                        "imageResponseCard": null
                    },
                    "variations": null
                }
            ]
        }
    },
    "intentClosingSetting": null,
    "inputContexts": null,
    "outputContexts": null,
    "kendraConfiguration": null,
    "dialogCodeHook": null,
    "fulfillmentCodeHook": null,
    "slotPriorities": [
        {
            "slotName": "DriverAge",
            "priority": 4
        },
        {
            "slotName": "PickUpDate",
            "priority": 2
        },
        {
            "slotName": "ReturnDate",
            "priority": 3
        },
        {
            "slotName": "PickUpCity",
            "priority": 1
        },
        {
            "slotName": "CarType",
            "priority": 5
        }
    ]
```

```
}
```

Structure du fichier Slot

Le fichier d'emplacement contient les informations de configuration d'un emplacement dans une intention. Le fichier .zip contient un fichier d'emplacement pour chaque emplacement défini pour une intention dans un environnement régional spécifique.

L'exemple suivant est la structure JSON d'un emplacement qui permet au client de choisir le type de voiture qu'il souhaite louer dans l'BookCarintention de l'BookTripexemple de bot. Pour un exemple complet de la structure JSON d'un slot, consultez l'CreateSlotopération.

```
{
    "name": "CarType",
    "identifier": "KDHJWNGZGC",
    "description": "Type of car being reserved.",
    "multipleValuesSetting": {
        "allowMutlipleValues": false
    },
    "slotTypeName": "CarTypeValues",
    "obfuscationSetting": null,
    "slotConstraint": "Required",
    "defaultValueSpec": null,
    "slotValueElicitationSetting": {
        "promptSpecification": {
            "messageGroupList": [
                {
                    "message": {
                        "plainTextMessage": {
                            "value": "What type of car would you like to rent?  Our
 most popular options are economy, midsize, and luxury"
                        },
                        "ssmlMessage": null,
                        "customPayload": null,
                        "imageResponseCard": null
                    },
                    "variations": null
                }
            ],
            "maxRetries": 2
        },
        "sampleValueElicitingUtterances": null,
```

```
        "waitAndContinueSpecification": null,
    }
}
```

L'exemple suivant montre la structure JSON d'un emplacement composite.

```
{
  "name": "CarType",
  "identifier": "KDHJWNGZGC",
  "description": "Type of car being reserved.",
  "multipleValuesSetting": {
      "allowMutlipleValues": false
  },
  "slotTypeName": "CarTypeValues",
  "obfuscationSetting": null,
  "slotConstraint": "Required",
  "defaultValueSpec": null,
  "slotValueElicitationSetting": {
      "promptSpecification": {
          "messageGroupList": [
              {
                  "message": {
                      "plainTextMessage": {
                          "value": "What type of car would you like to rent?  Our most
 popular options are economy, midsize, and luxury"
                      },
                      "ssmlMessage": null,
                      "customPayload": null,
                      "imageResponseCard": null
                  },
                  "variations": null
              }
          ],
          "maxRetries": 2
      },
      "sampleValueElicitingUtterances": null,
      "waitAndContinueSpecification": null,
  },
  "subSlotSetting": {
    "slotSpecifications": {
      "firstname": {
        "valueElicitationSetting": {
          "promptSpecification": {
```

```
            "allowInterrupt": false,
            "messageGroupsList": [
              {
                "message": {
                  "imageResponseCard": null,
                  "ssmlMessage": null,
                  "customPayload": null,
                  "plainTextMessage": {
                    "value": "please provide firstname"
                  }
                },
                "variations": null
              }
            ],
            "maxRetries": 2,
            "messageSelectionStrategy": "Random"
          },
          "defaultValueSpecification": null,
          "sampleUtterances": [
            {
              "utterance": "my name is {firstName}"
            }
          ],
          "waitAndContinueSpecification": null
        },
        "slotTypeId": "AMAZON.FirstName"
      },
      "eyeColor": {
        "valueElicitationSetting": {
          "promptSpecification": {
            "allowInterrupt": false,
            "messageGroupsList": [
              {
                "message": {
                  "imageResponseCard": null,
                  "ssmlMessage": null,
                  "customPayload": null,
                  "plainTextMessage": {
                    "value": "please provide eye color"
                  }
                },
                "variations": null
              }
            ],
```

```
            "maxRetries": 2,
            "messageSelectionStrategy": "Random"
          },
          "defaultValueSpecification": null,
          "sampleUtterances": [
            {
              "utterance": "eye color is {eyeColor}"
            },
            {
              "utterance": "I have eyeColor eyes"
            }
          ],
          "waitAndContinueSpecification": null
        },
        "slotTypeId": "7FEVCB2PQE"
      }
    },
    "expression": "(firstname OR eyeColor)"
  }
}
```

Structure de fichier de type slot

Le fichier de type d'emplacement contient les informations de configuration pour un type d'emplacement personnalisé utilisé dans une langue ou un environnement régional. Le fichier .zip contient un fichier de type d'emplacement pour chaque type d'emplacement personnalisé dans un environnement régional spécifique.

Voici la structure JSON du type d'emplacement qui répertorie les types de voitures disponibles dans l'BookTripexemple de bot. Pour un exemple complet de la structure JSON d'un type de slot, consultez l'CreateSlotTypeopération.

```
{
    "name": "CarTypeValues",
    "identifier": "T1YUHGD9ZR",
    "description": "Enumeration representing possible types of cars available for
 hire",
    "slotTypeValues": [{
        "synonyms": null,
        "sampleValue": {
            "value": "economy"
        }
```

```
    }, {
        "synonyms": null,
        "sampleValue": {
            "value": "standard"
        }
    }, {
        "synonyms": null,
        "sampleValue": {
            "value": "midsize"
        }
    }, {
        "synonyms": null,
        "sampleValue": {
            "value": "full size"
        }
    }, {
        "synonyms": null,
        "sampleValue": {
            "value": "luxury"
        }
    }, {
        "synonyms": null,
        "sampleValue": {
            "value": "minivan"
        }
    }],
    "parentSlotTypeSignature": null,
    "valueSelectionSetting": {
        "resolutionStrategy": "TOP_RESOLUTION",
        "advancedRecognitionSetting": {
            "audioRecognitionStrategy": "UseSlotValuesAsCustomVocabulary"
        },
        "regexFilter": null
    }
}
```

L'exemple suivant montre la structure JSON d'un type d'emplacement composite.

```
{
  "name": "CarCompositeType",
  "identifier": "TPA3CC9V",
  "description": null,
  "slotTypeValues": null,
```

```
  "parentSlotTypeSignature": null,
  "valueSelectionSetting": {
    "regexFilter": null,
    "resolutionStrategy": "CONCATENATION"
  },
  "compositeSlotTypeSetting": {
    "subSlots": [
      {
        "name": "model",
        "slotTypeId": "MODELTYPEID" # custom slot type Id for model
      },
      {
        "name": "city",
        "slotTypeId": "AMAZON.City"
      },
      {
        "name": "country",
        "slotTypeId": "AMAZON.Country"
      },
      {
        "name": "make",
        "slotTypeId": "MAKETYPEID" # custom slot type Id for make
      }
    ]
  }
}
```

Voici un type d'emplacement qui utilise une grammaire personnalisée pour comprendre les énoncés du client. Pour plus d'informations, veuillez consulter Type d'emplacement de grammaire.

```
{
  "name": "custom_grammar",
  "identifier": "7KEAQIQKPX",
  "description": "Slot type using a custom grammar",
  "slotTypeValues": null,
  "parentSlotTypeSignature": null,
  "valueSelectionSetting": null,
  "externalSourceSetting": {
    "grammarSlotTypeSetting": {
      "source": {
        "kmsKeyArn": "arn:aws:kms:Region:123456789012:alias/customer-grxml-key",
        "s3BucketName": "grxml-test",
        "s3ObjectKey": "grxml_files/grammar.grxml"
```

```
      }
    }
  }
}
```

Structure de fichier de vocabulaire personnalisée.

Le fichier de vocabulaire personnalisé contient les entrées d'un vocabulaire personnalisé pour une langue ou une région donnée. Le fichier .zip contient un fichier de vocabulaire personnalisé pour chaque langue associée à un vocabulaire personnalisé.

Ce qui suit est un fichier de vocabulaire personnalisé pour un robot qui prend des commandes au restaurant. Il y a un fichier par langue dans le bot.

```
{
    "customVocabularyItems": [
        {
            "weight": 3,
            "phrase": "wafers"
        },
        {
            "weight": null,
            "phrase": "extra large"
        },
        {
            "weight": null,
            "phrase": "cremini mushroom soup"
        },
        {
            "weight": null,
            "phrase": "ramen"
        },
        {
            "weight": null,
            "phrase": "orzo"
        }
    ]
}
```

Étiquetage des ressources

Pour vous aider à gérer vos robots et alias de bot Amazon Lex V2, vous pouvez attribuer des métadonnées à chaque ressource sous forme d'identifications. Une balise est une étiquette que vous affectez à une ressource AWS. Chaque balise se compose d'une clé et d'une valeur.

Les balises vous permettent de classer vosAWS ressources de différentes manières, par objectif, par propriétaire ou par application. Les balises vous aident à :

- Identifier et organiser vos ressources AWS. De nombreusesAWS ressources prennent en charge le balisage. Vous pouvez donc attribuer la même balise à des ressources dans différents services afin d'indiquer que les ressources sont identiques. Par exemple, vous pouvez baliser un bot et les fonctions Lambda qu'il utilise avec la même balise.
- Répartir les coûts. Vous activez ces balises sur le tableau de bord AWS Billing and Cost Management. AWS utilise les balises pour classer vos coûts et pour vous fournir un rapport mensuel de répartition des coûts. Pour Amazon Lex V2, vous pouvez répartir les coûts pour chaque alias à l'aide de balises spécifiques à l'alias. Pour de plus amples informations, veuillez consulter Utilisation des identifications d’allocation des coûts dans le Guide de l'utilisateur AWS Billing and Cost Management.
- Contrôler l'accès à vos ressources . Vous pouvez utiliser des balises avec Amazon Lex V2 pour créer des politiques visant à contrôler l'accès aux ressources Amazon Lex V2. Ces politiques peuvent être associées à un rôle ou à un utilisateur IAM afin de permettre un contrôle d'accès basé sur des balises.

Vous pouvez gérer les balises à l'aide de laAWS Management ConsoleAWS Command Line Interface, de la ou de l'API Amazon Lex V2.

Identification de vos ressources

Si vous utilisez la console Amazon Lex V2, vous pouvez baliser les ressources lorsque vous les créez, ou vous pouvez ajouter les balises ultérieurement. Vous pouvez également utiliser la console pour mettre à jour ou supprimer des balises existantes.

Si vous utilisez l'AWS CLIAPI Amazon Lex V2, vous devez effectuer les opérations suivantes pour gérer les balises de votre ressource :

- CreateBotet CreateBotAlias— appliquez des balises lorsque vous créez un bot ou un alias de bot.

- ListTagsForResource— affiche les balises associées à une ressource.
- TagResource— ajoute et modifie des balises sur une ressource existante.
- UntagResource— supprime des balises d'une ressource.

Les ressources suivantes dans Amazon Lex V2 prennent en charge le balisage :

- Bots — utilisez un Amazon Resource Name (ARN) comme suit :
  - `arn:aws:lex:${Region}:${account}:bot/${bot-id}`
- Alias de bot : utilisez un ARN comme celui-ci :
  - `arn:aws:lex:${Region}:${account}:bot-alias/${bot-id}/${bot-alias-id}`

Les`bot-alias-id` valeurs`bot-id` et sont des chaînes alphanumériques en majuscules de 10 caractères.

Restrictions liées aux étiquettes

Les restrictions de base suivantes s'appliquent aux balises sur les ressources Amazon Lex V2 :

- Nombre maximum de clés : 50 à l'aide de la console, 200 à l'aide de l'API
- Longueur maximale de la clé - 128 caractères
- Longueur maximale de la valeur - 256 caractères
- Caractères valides pour les clés et valeurs — a-z, A-Z, 0-9, espace et les caractères suivants : _. : / =+- et @
- Les clés et les valeurs sont sensibles à la casse.
- N'utilisez pas`aws:` comme préfixe pour les clés, seul peutAWS utiliser cette valeur

Ressources de balisage (console)

Vous pouvez utiliser la console pour gérer les balises d'un bot ou d'un alias de bot. Vous pouvez ajouter des balises lorsque vous créez la ressource, ou vous pouvez ajouter, modifier ou supprimer des balises de ressources existantes.

Pour ajouter une balise lorsque vous créez un bot

1. Connectez-vous à l'AWS Management Consoleet ouvrez la console Amazon Lex à l'adresse https://console.aws.amazon.com/lex/.
2. Choisissez Create bot.
3. Dans la section Paramètres avancés de Configurer les paramètres du bot, choisissez Ajouter un nouveau tag. Vous pouvez ajouter des balises au bot et à l'`TestBotAlias`alias.
4. Choisissez Suivant pour continuer à créer votre bot.

Pour ajouter une balise lorsque vous créez un alias de bot

1. Connectez-vous à l'AWS Management Consoleet ouvrez la console Amazon Lex à l'adresse https://console.aws.amazon.com/lex/.
2. Choisissez le bot auquel vous souhaitez ajouter l'alias de bot.
3. Dans le menu de gauche, choisissez Alias, puis choisissez Créer un alias.
4. Dans Informations générales, choisissez Ajouter une nouvelle étiquette à partir des balises.
5. Sélectionnez Create (Créer).

Pour ajouter, supprimer ou modifier une balise sur un bot existant

1. Connectez-vous à l'AWS Management Consoleet ouvrez la console Amazon Lex à l'adresse https://console.aws.amazon.com/lex/.
2. Choisissez le robot que vous souhaitez modifier.
3. Dans le menu de gauche, choisissez Paramètres, puis Modifier.
4. Dans Tags, apportez vos modifications.
5. Choisissez Enregistrer pour enregistrer vos modifications dans le bot.

Pour ajouter, supprimer ou modifier une étiquette sur un alias existant

1. Connectez-vous à l'AWS Management Consoleet ouvrez la console Amazon Lex à l'adresse https://console.aws.amazon.com/lex/.
2. Choisissez le robot que vous souhaitez modifier.
3. Dans le menu de gauche, choisissez Alias, puis dans la liste des alias, choisissez l'alias à modifier.

4. Dans Détails de l'alias, dans Balises, choisissez Modifier les balises.
5. Dans Gérer les balises, apportez vos modifications.
6. Choisissez Enregistrer pour enregistrer les modifications apportées à l'alias.

Sécurité dans Amazon Lex V2

La sécurité du cloud AWS est la priorité absolue. En tant que AWS client, vous bénéficiez de centres de données et d'architectures réseau conçus pour répondre aux exigences des entreprises les plus sensibles en matière de sécurité.

La sécurité est une responsabilité partagée entre vous AWS et vous. Le modèle de responsabilité partagée décrit cela comme la sécurité du cloud et la sécurité dans le cloud :

- Sécurité du cloud : AWS est chargée de protéger l'infrastructure qui exécute les AWS services dans le AWS cloud. AWS vous fournit également des services que vous pouvez utiliser en toute sécurité. Des auditeurs tiers testent et vérifient régulièrement l'efficacité de notre sécurité dans le cadre des programmes de AWS conformité Programmes de de conformité. Pour en savoir plus sur les programmes de conformité qui s'appliquent à Amazon Lex V2, consultez la section Services AWS concernés par programme de conformité .
- Sécurité dans le cloud — Votre responsabilité est déterminée par le AWS service que vous utilisez. Vous êtes également responsable d'autres facteurs, y compris de la sensibilité de vos données, des exigences de votre entreprise, ainsi que de la législation et de la réglementation applicables.

Cette documentation vous aide à comprendre comment appliquer le modèle de responsabilité partagée lors de l'utilisation d'Amazon Lex V2. Les rubriques suivantes expliquent comment configurer Amazon Lex V2 pour atteindre vos objectifs de sécurité et de conformité. Vous apprendrez également à utiliser d'autres services AWS qui vous aident à surveiller et à sécuriser vos ressources Amazon Lex V2.

Rubriques

Protection des données dans Amazon Lex V2

Amazon Lex V2 est conforme au modèle de responsabilité AWS partagée, modèle de , qui inclut des réglementations et des directives relatives à la protection des données. AWS est chargé de protéger l'infrastructure mondiale qui gère tous les AWS services. AWS conserve le contrôle des données hébergées sur cette infrastructure, y compris les contrôles de configuration de sécurité pour le traitement du contenu client et des données personnelles. AWS les clients et les partenaires APN, agissant en tant que contrôleurs ou sous-traitants de données, sont responsables de toutes les données personnelles qu'ils placent dans le AWS Cloud.

À des fins de protection des données, nous vous recommandons de protéger les informations d'identification des AWS comptes et de configurer des comptes utilisateur individuels avec AWS Identity and Access Management (IAM), afin que chaque utilisateur ne dispose que des autorisations nécessaires pour accomplir ses tâches. Nous vous recommandons également de sécuriser vos données comme indiqué ci-dessous :

- Utilisez l’authentification multifactorielle (MFA) avec chaque compte.
- Utilisez le protocole SSL/TLS pour communiquer avec les ressources. AWS

- Configurez l'API et la journalisation de l'activité des utilisateurs avec AWS CloudTrail.
- Utilisez des solutions de AWS chiffrement, ainsi que tous les contrôles de sécurité par défaut au sein AWS des services.
- Utilisez des services de sécurité gérés avancés tels qu’Amazon Macie, qui contribuent à la découverte et à la sécurisation des données personnelles stockées dans Amazon S3.

Nous vous recommandons vivement de ne jamais placer d’informations identifiables sensibles, telles que les numéros de compte de vos clients, dans des champs de formulaire comme Nom. Cela inclut lorsque vous travaillez avec Amazon Lex V2 ou d'autres AWS services à l'aide de la console, de l'API ou AWS des SDK. AWS CLI Toutes les données que vous entrez dans Amazon Lex V2 ou dans d'autres services peuvent être récupérées pour être incluses dans les journaux de diagnostic. Lorsque vous fournissez une URL à un serveur externe, n’incluez pas les informations d’identification non chiffrées dans l’URL pour valider votre demande adressée au serveur.

Pour en savoir plus sur la protection des données, consultez le billet de blog Modèle de responsabilité partagée AWS et RGPD sur le Blog sur la sécurité d’AWS .

Chiffrement au repos

Amazon Lex V2 chiffre les énoncés des utilisateurs et les autres informations qu'il stocke.

Rubriques

Exemples d'énoncés

Lorsque vous développez un bot, vous pouvez fournir des exemples d'énoncé pour chaque intention et chaque option. Vous pouvez également fournir des valeurs personnalisées et synonymes pour les options. Ces informations sont cryptées au repos et ne sont utilisées que pour créer le bot et créer l'expérience client.

Attributs de session

Les attributs de session contiennent des informations spécifiques à l'application qui sont transmises entre Amazon Lex V2 et les applications clientes. Amazon Lex V2 transmet les attributs de session à toutes les AWS Lambda fonctions configurées pour un bot. Si une fonction Lambda ajoute ou met à jour des attributs de session, Amazon Lex V2 transmet les nouvelles informations à l'application cliente.

Les attributs de session sont conservés dans un magasin chiffré pour toute la durée de la session. Vous pouvez configurer la session de sorte qu'elle reste active pendant au moins 1 minute à 24 heures après le dernier énoncé utilisateur. Par défaut, la durée de la session est de 5 minutes.

Attributs de demande

Les attributs de demande contiennent des informations propres à la demande et s'appliquent uniquement à la demande en cours. Une application cliente utilise les attributs de requête pour envoyer des informations à Amazon Lex V2 lors de l'exécution.

Utilisez les attributs de demande pour transmettre des informations qui n'ont pas besoin de persister pendant la totalité de la session. Dans la mesure où les attributs de demande ne sont pas conservés entre les demandes, ils ne sont pas stockés.

Chiffrement en transit

Amazon Lex V2 utilise le protocole HTTPS pour communiquer avec votre application cliente. Il utilise le protocole HTTPS et AWS des signatures pour communiquer avec d'autres services, tels qu'Amazon Polly et pour AWS Lambda le compte de votre application.

Gestion des identités et des accès pour Amazon Lex V2

AWS Identity and Access Management (IAM) est un outil Service AWS qui permet à un administrateur de contrôler en toute sécurité l'accès aux AWS ressources. Les administrateurs IAM contrôlent qui peut être authentifié (connecté) et autorisé (autorisé) à utiliser les ressources Amazon Lex V2. IAM est un Service AWS outil que vous pouvez utiliser sans frais supplémentaires.

Rubriques

Public ciblé

La façon dont vous utilisez AWS Identity and Access Management (IAM) varie en fonction du travail que vous effectuez dans Amazon Lex V2.

Utilisateur du service : si vous utilisez le service Amazon Lex V2 pour effectuer votre travail, votre administrateur vous fournit les informations d'identification et les autorisations dont vous avez besoin. Au fur et à mesure que vous utilisez de plus en plus de fonctionnalités d'Amazon Lex V2 dans le cadre de votre travail, il se peut que vous ayez besoin d'autorisations supplémentaires. En comprenant bien la gestion des accès, vous saurez demander les autorisations appropriées à

votre administrateur. Si vous ne pouvez pas accéder à une fonctionnalité dans Amazon Lex V2, consultezRésolution des problèmes d'identité et d'accès à Amazon Lex V2.

Administrateur du service — Si vous êtes responsable des ressources Amazon Lex V2 au sein de votre entreprise, vous avez probablement un accès complet à Amazon Lex V2. C'est à vous de déterminer les fonctionnalités et les ressources d'Amazon Lex V2 auxquelles les utilisateurs de votre service doivent accéder. Vous devez ensuite soumettre les demandes à votre administrateur IAM pour modifier les autorisations des utilisateurs de votre service. Consultez les informations sur cette page pour comprendre les concepts de base d'IAM. Pour en savoir plus sur la manière dont votre entreprise peut utiliser IAM avec Amazon Lex V2, consultezComment Amazon Lex V2 fonctionne avec IAM.

Administrateur IAM : si vous êtes administrateur IAM, vous souhaiterez peut-être en savoir plus sur la manière dont vous pouvez rédiger des politiques pour gérer l'accès à Amazon Lex V2. Pour consulter des exemples de politiques basées sur l'identité Amazon Lex V2 que vous pouvez utiliser dans IAM, consultez. Exemples de politiques basées sur l'identité pour Amazon Lex V2

Authentification par des identités

L'authentification est la façon dont vous vous connectez à AWS l'aide de vos informations d'identification. Vous devez être authentifié (connecté à AWS) en tant qu'utilisateur IAM ou en assumant un rôle IAM. Utilisateur racine d'un compte AWS

Vous pouvez vous connecter en AWS tant qu'identité fédérée en utilisant les informations d'identification fournies par le biais d'une source d'identité. AWS IAM Identity Center Les utilisateurs (IAM Identity Center), l'authentification unique de votre entreprise et vos informations d'identification Google ou Facebook sont des exemples d'identités fédérées. Lorsque vous vous connectez avec une identité fédérée, votre administrateur aura précédemment configuré une fédération d'identités avec des rôles IAM. Lorsque vous accédez à AWS l'aide de la fédération, vous assumez indirectement un rôle.

Selon le type d'utilisateur que vous êtes, vous pouvez vous connecter au portail AWS Management Console ou au portail AWS d'accès. Pour plus d'informations sur la connexion à AWS, consultez la section Comment vous connecter à votre compte Compte AWS dans le guide de Connexion à AWS l'utilisateur.

Si vous y accédez AWS par programmation, AWS fournit un kit de développement logiciel (SDK) et une interface de ligne de commande (CLI) pour signer cryptographiquement vos demandes à l'aide de vos informations d'identification. Si vous n'utilisez pas d' AWS outils, vous devez signer vous-

même les demandes. Pour plus d'informations sur l'utilisation de la méthode recommandée pour signer vous-même les demandes, consultez la section Signature des demandes AWS d'API dans le guide de l'utilisateur IAM.

Quelle que soit la méthode d'authentification que vous utilisez, vous devrez peut-être fournir des informations de sécurité supplémentaires. Par exemple, il vous AWS recommande d'utiliser l'authentification multifactorielle (MFA) pour renforcer la sécurité de votre compte. Pour en savoir plus, veuillez consulter Multi-factor authentication (Authentification multifactorielle) dans le Guide de l'utilisateur AWS IAM Identity Center et Utilisation de l'authentification multifactorielle (MFA) dans l'interface AWS dans le Guide de l'utilisateur IAM.

Compte AWS utilisateur root

Lorsque vous créez un Compte AWS, vous commencez par une identité de connexion unique qui donne un accès complet à toutes Services AWS les ressources du compte. Cette identité est appelée utilisateur Compte AWS root et est accessible en vous connectant avec l'adresse e-mail et le mot de passe que vous avez utilisés pour créer le compte. Il est vivement recommandé de ne pas utiliser l'utilisateur racine pour vos tâches quotidiennes. Protégez vos informations d'identification d'utilisateur racine et utilisez-les pour effectuer les tâches que seul l'utilisateur racine peut effectuer. Pour obtenir la liste complète des tâches qui vous imposent de vous connecter en tant qu'utilisateur root, consultez Tâches nécessitant des informations d'identification d'utilisateur root dans le Guide de l'utilisateur IAM.

Identité fédérée

La meilleure pratique consiste à obliger les utilisateurs humains, y compris ceux qui ont besoin d'un accès administrateur, à utiliser la fédération avec un fournisseur d'identité pour accéder à l'aide Services AWS d'informations d'identification temporaires.

Une identité fédérée est un utilisateur de l'annuaire des utilisateurs de votre entreprise, d'un fournisseur d'identité Web AWS Directory Service, du répertoire Identity Center ou de tout utilisateur qui y accède à l'aide des informations d'identification fournies Services AWS par le biais d'une source d'identité. Lorsque des identités fédérées y accèdent Comptes AWS, elles assument des rôles, qui fournissent des informations d'identification temporaires.

Pour une gestion des accès centralisée, nous vous recommandons d'utiliser AWS IAM Identity Center. Vous pouvez créer des utilisateurs et des groupes dans IAM Identity Center, ou vous pouvez vous connecter et synchroniser avec un ensemble d'utilisateurs et de groupes dans votre propre source d'identité afin de les utiliser dans toutes vos applications Comptes AWS et applications. Pour

obtenir des informations sur IAM Identity Center, consultez Qu'est-ce que IAM Identity Center ? dans le Guide de l'utilisateur AWS IAM Identity Center .

Utilisateurs et groupes IAM

Un utilisateur IAM est une identité au sein de vous Compte AWS qui possède des autorisations spécifiques pour une seule personne ou application. Dans la mesure du possible, nous vous recommandons de vous appuyer sur des informations d'identification temporaires plutôt que de créer des utilisateurs IAM ayant des informations d'identification à long terme tels que les clés d'accès. Toutefois, si certains cas d'utilisation spécifiques nécessitent des informations d'identification à long terme avec les utilisateurs IAM, nous vous recommandons de faire pivoter les clés d'accès. Pour plus d'informations, consultez Rotation régulière des clés d'accès pour les cas d'utilisation nécessitant des informations d'identification dans le Guide de l'utilisateur IAM.

Un groupe IAM est une identité qui concerne un ensemble d'utilisateurs IAM. Vous ne pouvez pas vous connecter en tant que groupe. Vous pouvez utiliser les groupes pour spécifier des autorisations pour plusieurs utilisateurs à la fois. Les groupes permettent de gérer plus facilement les autorisations pour de grands ensembles d'utilisateurs. Par exemple, vous pouvez avoir un groupe nommé IAMAdmins et accorder à ce groupe les autorisations d'administrer des ressources IAM.

Les utilisateurs sont différents des rôles. Un utilisateur est associé de manière unique à une personne ou une application, alors qu'un rôle est conçu pour être endossé par tout utilisateur qui en a besoin. Les utilisateurs disposent d'informations d'identification permanentes, mais les rôles fournissent des informations d'identification temporaires. Pour en savoir plus, consultez Quand créer un utilisateur IAM (au lieu d'un rôle) dans le Guide de l'utilisateur IAM.

Rôles IAM

Un rôle IAM est une identité au sein de vous Compte AWS dotée d'autorisations spécifiques. Le concept ressemble à celui d'utilisateur IAM, mais le rôle IAM n'est pas associé à une personne en particulier. Vous pouvez assumer temporairement un rôle IAM dans le en AWS Management Console changeant de rôle. Vous pouvez assumer un rôle en appelant une opération d' AWS API AWS CLI ou en utilisant une URL personnalisée. Pour plus d'informations sur les méthodes d'utilisation des rôles, consultez Utilisation de rôles IAM dans le Guide de l'utilisateur IAM.

Les rôles IAM avec des informations d'identification temporaires sont utiles dans les cas suivants :

- Accès utilisateur fédéré – Pour attribuer des autorisations à une identité fédérée, vous créez un rôle et définissez des autorisations pour le rôle. Quand une identité externe s'authentifie,

l'identité est associée au rôle et reçoit les autorisations qui sont définies par celui-ci. Pour obtenir des informations sur les rôles pour la fédération, consultez Création d'un rôle pour un fournisseur d'identité tiers (fédération) dans le Guide de l'utilisateur IAM. Si vous utilisez IAM Identity Center, vous configurez un jeu d'autorisations. IAM Identity Center met en corrélation le jeu d'autorisations avec un rôle dans IAM afin de contrôler à quoi vos identités peuvent accéder après leur authentification. Pour plus d'informations sur les jeux d'autorisations, veuillez consulter la rubrique Jeux d'autorisations dans le Guide de l'utilisateur AWS IAM Identity Center .

- Autorisations d'utilisateur IAM temporaires : un rôle ou un utilisateur IAM peut endosser un rôle IAM pour profiter temporairement d'autorisations différentes pour une tâche spécifique.
- Accès intercompte : vous pouvez utiliser un rôle IAM pour permettre à un utilisateur (principal de confiance) d'un compte différent d'accéder aux ressources de votre compte. Les rôles constituent le principal moyen d'accorder l'accès intercompte. Toutefois, dans certains Services AWS cas, vous pouvez associer une politique directement à une ressource (au lieu d'utiliser un rôle comme proxy). Pour en savoir plus sur la différence entre les rôles et les politiques basées sur les ressources pour l'accès intercompte, consultez Différence entre les rôles IAM et les politiques basées sur les ressources dans le Guide de l'utilisateur IAM.
- Accès multiservices — Certains Services AWS utilisent des fonctionnalités dans d'autres Services AWS. Par exemple, lorsque vous effectuez un appel dans un service, il est courant que ce service exécute des applications dans Amazon EC2 ou stocke des objets dans Amazon S3. Un service peut le faire en utilisant les autorisations d'appel du principal, une fonction de service ou un rôle lié au service.
  - Sessions d'accès direct (FAS) : lorsque vous utilisez un utilisateur ou un rôle IAM pour effectuer des actions AWS, vous êtes considéré comme un mandant. Lorsque vous utilisez certains services, l'action que vous effectuez est susceptible de lancer une autre action dans un autre service. FAS utilise les autorisations du principal appelant et Service AWS, associées Service AWS à la demande, pour adresser des demandes aux services en aval. Les demandes FAS ne sont effectuées que lorsqu'un service reçoit une demande qui nécessite des interactions avec d'autres personnes Services AWS ou des ressources pour être traitée. Dans ce cas, vous devez disposer d'autorisations nécessaires pour effectuer les deux actions. Pour plus de détails sur la politique relative à la transmission de demandes FAS, consultez Sessions de transmission d'accès.
  - Fonction du service : il s'agit d'un rôle IAM attribué à un service afin de réaliser des actions en votre nom. Un administrateur IAM peut créer, modifier et supprimer une fonction du service à partir d'IAM. Pour plus d'informations, consultez Création d'un rôle pour la délégation d'autorisations à un Service AWS dans le Guide de l'utilisateur IAM.

- Rôle lié à un service — Un rôle lié à un service est un type de rôle de service lié à un. Service AWS Le service peut endosser le rôle afin d'effectuer une action en votre nom. Les rôles liés à un service apparaissent dans votre Compte AWS répertoire et appartiennent au service. Un administrateur IAM peut consulter, mais ne peut pas modifier, les autorisations concernant les rôles liés à un service.

- Applications exécutées sur Amazon EC2 : vous pouvez utiliser un rôle IAM pour gérer les informations d'identification temporaires pour les applications qui s'exécutent sur une instance EC2 et qui envoient des demandes d'API. AWS CLI AWS Cette solution est préférable au stockage des clés d'accès au sein de l'instance EC2. Pour attribuer un AWS rôle à une instance EC2 et le mettre à la disposition de toutes ses applications, vous devez créer un profil d'instance attaché à l'instance. Un profil d'instance contient le rôle et permet aux programmes qui s'exécutent sur l'instance EC2 d'obtenir des informations d'identification temporaires. Pour plus d'informations, consultez Utilisation d'un rôle IAM pour accorder des autorisations à des applications s'exécutant sur des instances Amazon EC2 dans le Guide de l'utilisateur IAM.

Pour savoir dans quel cas utiliser des rôles ou des utilisateurs IAM, consultez Quand créer un rôle IAM (au lieu d'un utilisateur) dans le Guide de l'utilisateur IAM.

Gestion des accès à l'aide de politiques

Vous contrôlez l'accès en AWS créant des politiques et en les associant à AWS des identités ou à des ressources. Une politique est un objet AWS qui, lorsqu'il est associé à une identité ou à une ressource, définit leurs autorisations. AWS évalue ces politiques lorsqu'un principal (utilisateur, utilisateur root ou session de rôle) fait une demande. Les autorisations dans les politiques déterminent si la demande est autorisée ou refusée. La plupart des politiques sont stockées AWS sous forme de documents JSON. Pour plus d'informations sur la structure et le contenu des documents de politique JSON, consultez Présentation des politiques JSON dans le Guide de l'utilisateur IAM.

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

Par défaut, les utilisateurs et les rôles ne disposent d'aucune autorisation. Pour octroyer aux utilisateurs des autorisations d'effectuer des actions sur les ressources dont ils ont besoin, un administrateur IAM peut créer des politiques IAM. L'administrateur peut ensuite ajouter les politiques IAM aux rôles et les utilisateurs peuvent assumer les rôles.

Les politiques IAM définissent les autorisations d'une action, quelle que soit la méthode que vous utilisez pour exécuter l'opération. Par exemple, supposons que vous disposiez d'une politique qui autorise l'action `iam:GetRole`. Un utilisateur appliquant cette politique peut obtenir des informations sur le rôle à partir de AWS Management Console AWS CLI, de ou de l' AWS API.

Politiques basées sur l'identité

Les politiques basées sur l'identité sont des documents de politique d'autorisations JSON que vous pouvez attacher à une identité telle qu'un utilisateur, un Groupes d'utilisateurs IAM ou un rôle IAM. Ces politiques contrôlent quel type d'actions des utilisateurs et des rôles peuvent exécuter, sur quelles ressources et dans quelles conditions. Pour découvrir comment créer une politique basée sur l'identité, consultez Création de politiques IAM dans le Guide de l'utilisateur IAM.

Les politiques basées sur l'identité peuvent être classées comme des politiques en ligne ou des politiques gérées. Les politiques en ligne sont intégrées directement à un utilisateur, groupe ou rôle. Les politiques gérées sont des politiques autonomes que vous pouvez associer à plusieurs utilisateurs, groupes et rôles au sein de votre Compte AWS. Les politiques gérées incluent les politiques AWS gérées et les politiques gérées par le client. Pour découvrir comment choisir entre une politique gérée et une politique en ligne, consultez Choix entre les politiques gérées et les politiques en ligne dans le Guide de l'utilisateur IAM.

politiques basées sur les ressources

Les politiques basées sur les ressources sont des documents de politique JSON que vous attachez à une ressource. Des politiques basées sur les ressources sont, par exemple, les politiques de confiance de rôle IAM et des politiques de compartiment Amazon S3. Dans les services qui sont compatibles avec les politiques basées sur les ressources, les administrateurs de service peuvent les utiliser pour contrôler l'accès à une ressource spécifique. Pour la ressource dans laquelle se trouve la politique, cette dernière définit quel type d'actions un principal spécifié peut effectuer sur cette ressource et dans quelles conditions. Vous devez spécifier un principal dans une politique basée sur les ressources. Les principaux peuvent inclure des comptes, des utilisateurs, des rôles, des utilisateurs fédérés ou. Services AWS

Les politiques basées sur les ressources sont des politiques en ligne situées dans ce service. Vous ne pouvez pas utiliser les politiques AWS gérées par IAM dans une stratégie basée sur les ressources.

Listes de contrôle d’accès (ACL)

Les listes de contrôle d’accès (ACL) vérifie quels principaux (membres de compte, utilisateurs ou rôles) ont l’autorisation d’accéder à une ressource. Les listes de contrôle d’accès sont similaires aux politiques basées sur les ressources, bien qu’elles n’utilisent pas le format de document de politique JSON.

Amazon S3 et Amazon VPC sont des exemples de services qui prennent en charge les ACL. AWS WAF Pour en savoir plus sur les listes de contrôle d’accès, consultez Présentation des listes de contrôle d’accès (ACL) dans le Guide du développeur Amazon Simple Storage Service.

Autres types de politique

AWS prend en charge d'autres types de politiques moins courants. Ces types de politiques peuvent définir le nombre maximum d’autorisations qui vous sont accordées par des types de politiques plus courants.

- Limite d’autorisations – Une limite des autorisations est une fonctionnalité avancée dans laquelle vous définissez le nombre maximal d’autorisations qu’une politique basée sur l’identité peut accorder à une entité IAM (utilisateur IAM ou rôle IAM). Vous pouvez définir une limite d’autorisations pour une entité. Les autorisations qui en résultent représentent la combinaison des politiques basées sur l’identité d’une entité et de ses limites d’autorisation. Les politiques basées sur les ressources qui spécifient l’utilisateur ou le rôle dans le champ `Principal` ne sont pas limitées par les limites d’autorisations. Un refus explicite dans l’une de ces politiques remplace l’autorisation. Pour plus d’informations sur les limites d’autorisations, consultez Limites d’autorisations pour des entités IAM dans le Guide de l’utilisateur IAM.
- Politiques de contrôle des services (SCP) — Les SCP sont des politiques JSON qui spécifient les autorisations maximales pour une organisation ou une unité organisationnelle (UO) dans. AWS Organizations AWS Organizations est un service permettant de regrouper et de gérer de manière centralisée Comptes AWS les multiples propriétés de votre entreprise. Si vous activez toutes les fonctions d’une organisation, vous pouvez appliquer les politiques de contrôle de service (SCP) à l’un ou à l’ensemble de vos comptes. Le SCP limite les autorisations pour les entités figurant dans les comptes des membres, y compris chacune Utilisateur racine d'un compte AWS d'entre elles. Pour plus d’informations sur les organisations et les SCP, consultez Fonctionnement des SCP dans le Guide de l’utilisateur AWS Organizations .
- politiques de séance : les politiques de séance sont des politiques avancées que vous utilisez en tant que paramètre lorsque vous créez par programmation une séance temporaire pour un rôle ou un utilisateur fédéré. Les autorisations de la séance obtenue sont une combinaison des politiques

basées sur l'identité de l'utilisateur ou du rôle et des politiques de séance. Les autorisations peuvent également provenir d'une politique basée sur les ressources. Un refus explicite dans l'une de ces politiques remplace l'autorisation. Pour plus d'informations, consultez Politiques de séance dans le Guide de l'utilisateur IAM.

Plusieurs types de politique

Lorsque plusieurs types de politiques s'appliquent à la requête, les autorisations obtenues sont plus compliquées à comprendre. Pour savoir comment AWS déterminer s'il faut autoriser une demande lorsque plusieurs types de politiques sont impliqués, consultez la section Logique d'évaluation des politiques dans le guide de l'utilisateur IAM.

Comment Amazon Lex V2 fonctionne avec IAM

Avant d'utiliser IAM pour gérer l'accès à Amazon Lex V2, découvrez quelles fonctionnalités IAM peuvent être utilisées avec Amazon Lex V2.

Fonctionnalités IAM que vous pouvez utiliser avec Amazon Lex V2

Fonction IAM	Assistance avec Amazon Lex V2
Politiques basées sur l'identité	Oui
Politiques basées sur les ressources	Oui
Actions de politique	Oui
Ressources de politique	Oui
Clés de condition d'une politique	Non
ACL	Non
ABAC (étiquettes dans les politiques)	Oui
Informations d'identification temporaires	Non
Autorisations de principal	Oui
Fonctions de service	Oui

Fonction IAM	Assistance avec Amazon Lex V2
Rôles liés à un service	Partielle

Pour obtenir une vue d'ensemble de la façon dont Amazon Lex V2 et les autres AWS services fonctionnent avec la plupart des fonctionnalités IAM, consultez les AWS services compatibles avec IAM dans le guide de l'utilisateur IAM.

Politiques basées sur l'identité pour Amazon Lex V2

Prend en charge les politiques basées sur l’identité	Oui

Les politiques basées sur l’identité sont des documents de politique d’autorisations JSON que vous pouvez attacher à une identité telle qu’un utilisateur, un Groupes d’utilisateurs IAM ou un rôle IAM. Ces politiques contrôlent quel type d’actions des utilisateurs et des rôles peuvent exécuter, sur quelles ressources et dans quelles conditions. Pour découvrir comment créer une politique basée sur l’identité, veuillez consulter Création de politiques IAM dans le Guide de l’utilisateur IAM.

Avec les politiques IAM basées sur l’identité, vous pouvez spécifier des actions et ressources autorisées ou refusées, ainsi que les conditions dans lesquelles les actions sont autorisées ou refusées. Vous ne pouvez pas spécifier le principal dans une politique basée sur une identité car celle-ci s’applique à l’utilisateur ou au rôle auquel elle est attachée. Pour découvrir tous les éléments que vous utilisez dans une politique JSON, consultez Références des éléments de politique JSON IAM dans le Guide de l’utilisateur IAM.

Exemples de politiques basées sur l'identité pour Amazon Lex V2

Pour consulter des exemples de politiques basées sur l'identité Amazon Lex V2, consultez. Exemples de politiques basées sur l'identité pour Amazon Lex V2

Politiques basées sur les ressources dans Amazon Lex V2

Prend en charge les politiques basées sur les ressources	Oui

Les politiques basées sur les ressources sont des documents de politique JSON que vous attachez à une ressource. Des politiques basées sur les ressources sont, par exemple, les politiques de confiance de rôle IAM et des politiques de compartiment Amazon S3. Dans les services qui sont compatibles avec les politiques basées sur les ressources, les administrateurs de service peuvent les utiliser pour contrôler l'accès à une ressource spécifique. Pour la ressource dans laquelle se trouve la politique, cette dernière définit quel type d'actions un principal spécifié peut effectuer sur cette ressource et dans quelles conditions. Vous devez spécifier un principal dans une politique basée sur les ressources. Les principaux peuvent inclure des utilisateurs, des rôles, des utilisateurs fédérés ou des services AWS.

Vous ne pouvez pas utiliser de politiques entre comptes ou entre régions avec Amazon Lex. Si vous créez une politique pour une ressource avec un ARN entre comptes ou entre régions, Amazon Lex renvoie un message d'erreur.

Le service Amazon Lex prend en charge les politiques basées sur les ressources appelées politique de bot et politique d'alias de bot, qui sont associées à un bot ou à un alias de bot. Ces politiques définissent les principaux autorisés à effectuer des actions sur le bot ou l'alias du bot.

Les actions ne peuvent être utilisées que sur des ressources spécifiques. Par exemple, l'`UpdateBot`action ne peut être utilisée que sur les ressources du bot, l'`UpdateBotAlias`action ne peut être utilisée que sur les ressources d'alias du bot. Si vous spécifiez une action dans une politique qui ne peut pas être utilisée sur la ressource spécifiée dans la politique, Amazon Lex renvoie une erreur. Pour la liste des actions et des ressources avec lesquelles elles peuvent être utilisées, consultez le tableau suivant.

Action	Soutient la politique basée sur les ressources	Ressource
BuildBotLocale	Pris en charge	BotId
CreateBot	Non	
CreateBotAlias	Non	
CreateBotChannel [autorisation uniquement]	Pris en charge	BotId
CreateBotLocale	Pris en charge	BotId

Action	Soutient la politique basée sur les ressources	Ressource
CreateBotVersion	Pris en charge	BotId
CreateExport	Pris en charge	BotId
CreateIntent	Pris en charge	BotId
CreateResourcePolicy	Pris en charge	BotId, BotAliasId
CreateSlot	Pris en charge	BotId
CreateSlotType	Pris en charge	BotId
CreateUploadUrl	Non	
DeleteBot	Pris en charge	BotId, BotAliasId
DeleteBotAlias	Pris en charge	BotAliasId
DeleteBotChannel [autorisation uniquement]	Pris en charge	BotId
DeleteBotLocale	Pris en charge	BotId
DeleteBotVersion	Pris en charge	BotId
DeleteExport	Pris en charge	BotId
DeleteImport	Pris en charge	BotId
DeleteIntent	Pris en charge	BotId
DeleteResourcePolicy	Pris en charge	BotId, BotAliasId
DeleteSession	Pris en charge	BotAliasId
DeleteSlot	Pris en charge	BotId
DeleteSlotType	Pris en charge	BotId

Action	Soutient la politique basée sur les ressources	Ressource
DescribeBot	Pris en charge	BotId
DescribeBotAlias	Pris en charge	BotAliasId
DescribeBotChannel [autorisation uniquement]	Pris en charge	BotId
DescribeBotLocale	Pris en charge	BotId
DescribeBotVersion	Pris en charge	BotId
DescribeExport	Pris en charge	BotId
DescribeImport	Pris en charge	BotId
DescribeIntent	Pris en charge	BotId
DescribeResourcePolicy	Pris en charge	BotId, BotAliasId
DescribeSlot	Pris en charge	BotId
DescribeSlotType	Pris en charge	BotId
GetSession	Pris en charge	BotAliasId
ListBotAliases	Pris en charge	BotId
ListBotChannels [autorisation uniquement]	Pris en charge	BotId
ListBotLocales	Pris en charge	BotId
ListBots	Non	
ListBotVersions	Pris en charge	BotId
ListBuiltInIntents	Non	
ListBuiltIntSlotTypes	Non	

Action	Soutient la politique basée sur les ressources	Ressource
ListExports	Non	
ListImports	Non	
ListIntents	Pris en charge	BotId
ListSlots	Pris en charge	BotId
ListSlotTypes	Pris en charge	BotId
PutSession	Pris en charge	BotAliasId
RecognizeText	Pris en charge	BotAliasId
RecognizeUtterance	Pris en charge	BotAliasId
StartConversation	Pris en charge	BotAliasId
StartImport	Pris en charge	BotId, BotAliasId
TagResource	Non	
UpdateBot	Pris en charge	BotId
UpdateBotAlias	Pris en charge	BotAliasId
UpdateBotLocale	Pris en charge	BotId
UpdateBotVersion	Pris en charge	BotId
UpdateExport	Pris en charge	BotId
UpdateIntent	Pris en charge	BotId
UpdateResourcePolicy	Pris en charge	BotId, BotAliasId
UpdateSlot	Pris en charge	BotId
UpdateSlotType	Pris en charge	BotId

Action	Soutient la politique basée sur les ressources	Ressource
UntagResource	Non	

Pour savoir comment associer une politique basée sur les ressources à un bot ou à un alias de bot, consultez. Exemples de politiques basées sur les ressources pour Amazon Lex V2

Exemples de politiques basées sur les ressources dans Amazon Lex V2

Pour consulter des exemples de politiques basées sur les ressources Amazon Lex V2, consultez. Exemples de politiques basées sur les ressources pour Amazon Lex V2

Actions politiques pour Amazon Lex V2

Prend en charge les actions de politique	Oui

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément `Action` d'une politique JSON décrit les actions que vous pouvez utiliser pour autoriser ou refuser l'accès à une politique. Les actions de stratégie portent généralement le même nom que l'opération AWS d'API associée. Il existe quelques exceptions, telles que les actions avec autorisations uniquement qui n'ont pas d'opération API correspondante. Certaines opérations nécessitent également plusieurs actions dans une politique. Ces actions supplémentaires sont nommées actions dépendantes.

Intégration d'actions dans une stratégie afin d'accorder l'autorisation d'exécuter les opérations associées.

Pour consulter la liste des actions Amazon Lex V2, consultez la section Actions définies par Amazon Lex V2 dans le Service Authorization Reference.

Les actions politiques dans Amazon Lex V2 utilisent le préfixe suivant avant l'action :

```
lex
```

Pour indiquer plusieurs actions dans une seule déclaration, séparez-les par des virgules.

```
"Action": [
      "lex:action1",
      "lex:action2"
         ]
```

Pour consulter des exemples de politiques basées sur l'identité Amazon Lex V2, consultez. Exemples de politiques basées sur l'identité pour Amazon Lex V2

Ressources relatives aux politiques pour Amazon Lex V2

Prend en charge les ressources de politique	Oui

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément de politique JSON `Resource` indique le ou les objets pour lesquels l'action s'applique. Les instructions doivent inclure un élément `Resource` ou `NotResource`. Il est recommandé de définir une ressource à l'aide de son Amazon Resource Name (ARN). Vous pouvez le faire pour des actions qui prennent en charge un type de ressource spécifique, connu sous la dénomination autorisations de niveau ressource.

Pour les actions qui ne sont pas compatibles avec les autorisations de niveau ressource, telles que les opérations de liste, utilisez un caractère générique (*) afin d'indiquer que l'instruction s'applique à toutes les ressources.

```
"Resource": "*"
```

Pour consulter la liste des types de ressources Amazon Lex V2 et de leurs ARN, consultez la section Ressources définies par Amazon Lex V2 dans le Service Authorization Reference. Pour savoir avec quelles actions vous pouvez spécifier l'ARN de chaque ressource, consultez Actions définies par Amazon Lex V2.

Pour consulter des exemples de politiques basées sur l'identité Amazon Lex V2, consultez. Exemples de politiques basées sur l'identité pour Amazon Lex V2

Clés de conditions de politique pour Amazon Lex V2

Prend en charge les clés de condition de politique spécifiques au service	Non

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément `Condition` (ou le bloc `Condition`) vous permet de spécifier des conditions lorsqu'une instruction est appliquée. L'élément `Condition` est facultatif. Vous pouvez créer des expressions conditionnelles qui utilisent des opérateurs de condition, tels que les signes égal ou inférieur à, pour faire correspondre la condition de la politique aux valeurs de la demande.

Si vous spécifiez plusieurs éléments `Condition` dans une instruction, ou plusieurs clés dans un seul élément `Condition`, AWS les évalue à l'aide d'une opération `AND` logique. Si vous spécifiez plusieurs valeurs pour une seule clé de condition, AWS évalue la condition à l'aide d'une `OR` opération logique. Toutes les conditions doivent être remplies avant que les autorisations associées à l'instruction ne soient accordées.

Vous pouvez aussi utiliser des variables d'espace réservé quand vous spécifiez des conditions. Par exemple, vous pouvez accorder à un utilisateur IAM l'autorisation d'accéder à une ressource uniquement si elle est balisée avec son nom d'utilisateur IAM. Pour plus d'informations, consultez Éléments d'une politique IAM : variables et identifications dans le Guide de l'utilisateur IAM.

AWS prend en charge les clés de condition globales et les clés de condition spécifiques au service. Pour voir toutes les clés de condition AWS globales, voir les clés de contexte de condition AWS globales dans le guide de l'utilisateur IAM.

Pour consulter la liste des clés de condition Amazon Lex V2, consultez la section Clés de condition pour Amazon Lex V2 dans la référence d'autorisation de service. Pour savoir avec quelles actions et ressources vous pouvez utiliser une clé de condition, consultez Actions définies par Amazon Lex V2.

Pour consulter des exemples de politiques basées sur l'identité Amazon Lex V2, consultez. Exemples de politiques basées sur l'identité pour Amazon Lex V2

Listes de contrôle d'accès (ACL) dans Amazon Lex V2

Prend en charge les listes ACL	Non

Les listes de contrôle d'accès (ACL) vérifient quels principaux (membres de compte, utilisateurs ou rôles) ont l'autorisation d'accéder à une ressource. Les listes de contrôle d'accès sont similaires aux politiques basées sur les ressources, bien qu'elles n'utilisent pas le format de document de politique JSON.

Contrôle d'accès basé sur les attributs (ABAC) avec Amazon Lex V2

Prend en charge ABAC (étiquettes dans les politiques)	Oui

Le contrôle d'accès par attributs (ABAC) est une stratégie d'autorisation qui définit des autorisations en fonction des attributs. Dans AWS, ces attributs sont appelés balises. Vous pouvez associer des balises aux entités IAM (utilisateurs ou rôles) et à de nombreuses AWS ressources. L'étiquetage des entités et des ressources est la première étape d'ABAC. Vous concevez ensuite des politiques ABAC pour autoriser des opérations quand l'identification du principal correspond à celle de la ressource à laquelle il tente d'accéder.

L'ABAC est utile dans les environnements qui connaissent une croissance rapide et pour les cas où la gestion des politiques devient fastidieuse.

Pour contrôler l'accès basé sur des étiquettes, vous devez fournir les informations d'étiquette dans l'élément de condition d'une politique utilisant les clés de condition `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* ou `aws:TagKeys`.

Si un service prend en charge les trois clés de condition pour tous les types de ressources, alors la valeur pour ce service est Oui. Si un service prend en charge les trois clés de condition pour certains types de ressources uniquement, la valeur est Partielle.

Pour plus d'informations sur l'ABAC, consultez Qu'est-ce que le contrôle d'accès basé sur les attributs (ABAC) ? dans le Guide de l'utilisateur IAM. Pour accéder à un didacticiel décrivant les étapes de configuration de l'ABAC, consultez Utilisation du contrôle d'accès basé sur les attributs (ABAC) dans le Guide de l'utilisateur IAM.

Utilisation d'informations d'identification temporaires avec Amazon Lex V2

Prend en charge les informations d’identification temporaires	Non

Certains Services AWS ne fonctionnent pas lorsque vous vous connectez à l'aide d'informations d'identification temporaires. Pour plus d'informations, y compris celles qui Services AWS fonctionnent avec des informations d'identification temporaires, consultez Services AWS la section relative à l'utilisation d'IAM dans le guide de l'utilisateur d'IAM.

Vous utilisez des informations d'identification temporaires si vous vous connectez à l' AWS Management Console aide d'une méthode autre qu'un nom d'utilisateur et un mot de passe. Par exemple, lorsque vous accédez à AWS l'aide du lien d'authentification unique (SSO) de votre entreprise, ce processus crée automatiquement des informations d'identification temporaires. Vous créez également automatiquement des informations d’identification temporaires lorsque vous vous connectez à la console en tant qu’utilisateur, puis changez de rôle. Pour plus d’informations sur le changement de rôle, consultez Changement de rôle (console) dans le Guide de l’utilisateur IAM.

Vous pouvez créer manuellement des informations d'identification temporaires à l'aide de l' AWS API AWS CLI or. Vous pouvez ensuite utiliser ces informations d'identification temporaires pour y accéder AWS. AWS recommande de générer dynamiquement des informations d'identification temporaires au lieu d'utiliser des clés d'accès à long terme. Pour plus d’informations, consultez Informations d’identification de sécurité temporaires dans IAM.

Autorisations principales interservices pour Amazon Lex V2

Prend en charge les transmissions de sessions d’accès (FAS)	Oui

Lorsque vous utilisez un utilisateur ou un rôle IAM pour effectuer des actions AWS, vous êtes considéré comme un mandant. Lorsque vous utilisez certains services, l’action que vous effectuez est susceptible de lancer une autre action dans un autre service. FAS utilise les autorisations du principal appelant et Service AWS, associées Service AWS à la demande, pour adresser des demandes aux services en aval. Les demandes FAS ne sont effectuées que lorsqu'un service reçoit une demande qui nécessite des interactions avec d'autres personnes Services AWS ou des ressources pour être traitée. Dans ce cas, vous devez disposer d’autorisations nécessaires pour effectuer les deux

actions. Pour plus de détails sur une politique lors de la formulation de demandes FAS, consultez Transmission des sessions d'accès.

Rôles de service pour Amazon Lex V2

Prend en charge les fonctions de service	Oui

Une fonction du service est un rôle IAM qu'un service endosse pour accomplir des actions en votre nom. Un administrateur IAM peut créer, modifier et supprimer une fonction du service à partir d'IAM. Pour plus d'informations, consultez Création d'un rôle pour la délégation d'autorisations à un Service AWS dans le Guide de l'utilisateur IAM.

> **Warning**
>
> La modification des autorisations associées à un rôle de service peut perturber les fonctionnalités d'Amazon Lex V2. Modifiez les rôles de service uniquement lorsque Amazon Lex V2 fournit des instructions à cet effet.

Rôles liés à un service pour Amazon Lex V2

Prend en charge les rôles liés à un service	Partielle

Un rôle lié à un service est un type de rôle de service lié à un. Service AWS Le service peut endosser le rôle afin d'effectuer une action en votre nom. Les rôles liés à un service apparaissent dans votre Compte AWS répertoire et appartiennent au service. Un administrateur IAM peut consulter, mais ne peut pas modifier, les autorisations concernant les rôles liés à un service.

Pour plus d'informations sur la création ou la gestion des rôles liés à un service, consultez Services AWS qui fonctionnent avec IAM. Recherchez un service dans le tableau qui inclut un `Yes` dans la colonne Rôle lié à un service. Choisissez le lien Oui pour consulter la documentation du rôle lié à ce service.

Exemples de politiques basées sur l'identité pour Amazon Lex V2

Par défaut, les utilisateurs et les rôles ne sont pas autorisés à créer ou à modifier les ressources Amazon Lex V2. Ils ne peuvent pas non plus effectuer de tâches à l'aide de l'API AWS Management

Console, AWS Command Line Interface (AWS CLI) ou de AWS l'API. Pour octroyer aux utilisateurs des autorisations d'effectuer des actions sur les ressources dont ils ont besoin, un administrateur IAM doit créer des politiques IAM. L'administrateur peut ensuite ajouter les politiques IAM aux rôles et les utilisateurs peuvent assumer les rôles.

Pour apprendre à créer une politique basée sur l'identité IAM à l'aide de ces exemples de documents de politique JSON, consultez Création de politiques dans l'onglet JSON dans le Guide de l'utilisateur IAM.

Pour plus de détails sur les actions et les types de ressources définis par Amazon Lex V2, y compris le format des ARN pour chacun des types de ressources, consultez Actions, ressources et clés de condition pour Amazon Lex V2 dans la référence d'autorisation de service.

Rubriques

- Bonnes pratiques en matière de politiques
- Utilisation de la console Amazon Lex V2
- Autoriser les utilisateurs à ajouter des fonctions à un bot
- Autoriser les utilisateurs à ajouter des chaînes à un bot
- Permettre aux utilisateurs de créer et de mettre à jour des robots
- Autoriser les utilisateurs à utiliser le concepteur de Chatbot automatisé
- Autoriser les utilisateurs à utiliser une AWS KMS clé pour chiffrer et déchiffrer des fichiers
- Autoriser les utilisateurs à supprimer des robots
- Permettre aux utilisateurs d'avoir une conversation avec un bot
- Autoriser un utilisateur spécifique à gérer les politiques basées sur les ressources
- Autoriser un utilisateur à exporter des robots et des paramètres régionaux de robots
- Permettre à un utilisateur d'exporter un vocabulaire personnalisé
- Autoriser un utilisateur à importer des robots et des paramètres régionaux de robots
- Permettre à un utilisateur d'importer un vocabulaire personnalisé
- Autoriser un utilisateur à migrer un bot d'Amazon Lex vers Amazon Lex V2
- Autorisation accordée aux utilisateurs pour afficher leurs propres autorisations
- Permettre à un utilisateur de créer un flux de conversation avec le générateur de conversation visuel dans Amazon Lex V2
- Autoriser les utilisateurs à créer et à afficher des répliques de robots, mais pas à les supprimer

Bonnes pratiques en matière de politiques

Les politiques basées sur l'identité déterminent si quelqu'un peut créer, accéder ou supprimer des ressources Amazon Lex V2 dans votre compte. Ces actions peuvent entraîner des frais pour votre Compte AWS. Lorsque vous créez ou modifiez des politiques basées sur l'identité, suivez ces instructions et recommandations :

- Commencez AWS par les politiques gérées et passez aux autorisations du moindre privilège : pour commencer à accorder des autorisations à vos utilisateurs et à vos charges de travail, utilisez les politiques AWS gérées qui accordent des autorisations pour de nombreux cas d'utilisation courants. Ils sont disponibles dans votre Compte AWS. Nous vous recommandons de réduire davantage les autorisations en définissant des politiques gérées par les AWS clients spécifiques à vos cas d'utilisation. Pour de plus amples informations, consultez Politiques gérées AWS ou Politiques gérées AWS pour les activités professionnelles dans le Guide de l'utilisateur IAM.
- Accorder les autorisations de moindre privilège - Lorsque vous définissez des autorisations avec des politiques IAM, accordez uniquement les autorisations nécessaires à l'exécution d'une seule tâche. Pour ce faire, vous définissez les actions qui peuvent être entreprises sur des ressources spécifiques dans des conditions spécifiques, également appelées autorisations de moindre privilège. Pour plus d'informations sur l'utilisation de IAM pour appliquer des autorisations, consultez Politiques et autorisations dans IAM dans le Guide de l'utilisateur IAM.
- Utiliser des conditions dans les politiques IAM pour restreindre davantage l'accès - Vous pouvez ajouter une condition à vos politiques afin de limiter l'accès aux actions et aux ressources. Par exemple, vous pouvez écrire une condition de politique pour spécifier que toutes les demandes doivent être envoyées via SSL. Vous pouvez également utiliser des conditions pour accorder l'accès aux actions de service si elles sont utilisées par le biais d'un service spécifique Service AWS, tel que AWS CloudFormation. Pour plus d'informations, consultez Conditions pour éléments de politique JSON IAM dans le Guide de l'utilisateur IAM.
- Utilisez IAM Access Analyzer pour valider vos politiques IAM afin de garantir des autorisations sécurisées et fonctionnelles - IAM Access Analyzer valide les politiques nouvelles et existantes de manière à ce que les politiques IAM respectent le langage de politique IAM (JSON) et les bonnes pratiques IAM. IAM Access Analyzer fournit plus de 100 vérifications de politiques et des recommandations exploitables pour vous aider à créer des politiques sécurisées et fonctionnelles. Pour de plus amples informations, consultez Validation de politique IAM Access Analyzer dans le Guide de l'utilisateur IAM.
- Exiger l'authentification multifactorielle (MFA) : si vous avez un scénario qui nécessite des utilisateurs IAM ou un utilisateur root, activez l'authentification MFA pour une sécurité accrue.

Compte AWS Pour exiger le MFA lorsque des opérations d'API sont appelées, ajoutez des conditions MFA à vos politiques. Pour de plus amples informations, consultez Configuration de l'accès aux API protégé par MFA dans le Guide de l'utilisateur IAM.

Pour plus d'informations sur les bonnes pratiques dans IAM, consultez Bonnes pratiques de sécurité dans IAM dans le Guide de l'utilisateur IAM.

Utilisation de la console Amazon Lex V2

Pour accéder à la console Amazon Lex V2, vous devez disposer d'un ensemble minimal d'autorisations. Ces autorisations doivent vous permettre de répertorier et de consulter les informations relatives aux ressources Amazon Lex V2 présentes dans votre Compte AWS. Si vous créez une stratégie basée sur l'identité qui est plus restrictive que l'ensemble minimum d'autorisations requis, la console ne fonctionnera pas comme prévu pour les entités (utilisateurs ou rôles) tributaires de cette stratégie.

Il n'est pas nécessaire d'accorder des autorisations de console minimales aux utilisateurs qui appellent uniquement l'API AWS CLI ou l' AWS API. Autorisez plutôt l'accès à uniquement aux actions qui correspondent à l'opération d'API qu'ils tentent d'effectuer.

Pour garantir que les utilisateurs et les rôles peuvent toujours utiliser la console Amazon Lex V2, les utilisateurs doivent avoir accès à la console. Pour plus d'informations sur la création d'un utilisateur avec accès à la console, consultez la section Création d'un utilisateur IAM dans votre AWS compte dans le guide de l'utilisateur IAM.

Autoriser les utilisateurs à ajouter des fonctions à un bot

Cet exemple montre une politique qui permet aux utilisateurs d'IAM d'ajouter des autorisations Amazon Comprehend, d'analyse des sentiments et de requête Amazon Kendra à un bot Amazon Lex V2.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Id1",
            "Effect": "Allow",
            "Action": "iam:PutRolePolicy",
            "Resource": "arn:aws:iam::*:role/aws-service-role/lexv2.amazonaws.com/
AWSServiceRoleForLexV2Bots*"
```

```
        },
        {
            "Sid": "Id2",
            "Effect": "Allow",
            "Action": "iam:GetRolePolicy",
            "Resource": "arn:aws:iam::*:role/aws-service-role/lexv2.amazonaws.com/
AWSServiceRoleForLexV2Bots*"
        }
    ]
}
```

Autoriser les utilisateurs à ajouter des chaînes à un bot

Cet exemple est une politique qui permet aux utilisateurs IAM d'ajouter un canal de messagerie à un bot. Un utilisateur doit avoir mis en place cette politique avant de pouvoir déployer un bot sur une plateforme de messagerie.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Id1",
            "Effect": "Allow",
            "Action": "iam:PutRolePolicy",
            "Resource": "arn:aws:iam::*:role/aws-service-role/
channels.lexv2.amazonaws.com/AWSServiceRoleForLexV2Channels*"
        },
        {
            "Sid": "Id2",
            "Effect": "Allow",
            "Action": "iam:GetRolePolicy",
            "Resource": "arn:aws:iam::*:role/aws-service-role/
channels.lexv2.amazonaws.com/AWSServiceRoleForLexV2Channels*"
        }
    ]
}
```

Permettre aux utilisateurs de créer et de mettre à jour des robots

Cet exemple montre un exemple de politique qui permet aux utilisateurs IAM de créer et de mettre à jour n'importe quel bot. La politique inclut les autorisations permettant d'effectuer cette action sur la console ou à l'aide de l' AWS API AWS CLI or.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "lex:CreateBot",
        "lex:UpdateBot".
        "iam:PassRole"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:lex:Region:123412341234:bot/*]
    }
  ]
}
```

Autoriser les utilisateurs à utiliser le concepteur de Chatbot automatisé

Cet exemple montre un exemple de politique qui permet aux utilisateurs d'IAM d'exécuter le concepteur de Chatbot automatisé.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::<customer-bucket>/<bucketName>",
                # Resource should point to the bucket or an explicit folder.
                # Provide this to read the entire bucket
                "arn:aws:s3:::<customer-bucket>/<bucketName>/*",
                # Provide this to read a specifc folder
                "arn:aws:s3:::<customer-bucket>/<bucketName>/<pathFormat>/*"
            ]
        },
        {
            # Use this if your S3 bucket is encrypted with a KMS key.
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt"
```

```
            ],
            "Resource": [
                "arn:aws:kms:<Region>:<customerAccountId>:key/<kmsKeyId>"
            ]
    ]
}
```

Autoriser les utilisateurs à utiliser une AWS KMS clé pour chiffrer et déchiffrer des fichiers

Cet exemple montre un exemple de politique qui permet aux utilisateurs IAM d'utiliser une clé gérée par AWS KMS le client pour chiffrer et déchiffrer des données.

```
{
    "Version": "2012-10-17",
    "Id": "sample-policy",
    "Statement": [
        {
            "Sid": "Allow Lex access",
            "Effect": "Allow",
            "Principal": {
                "Service": "lexv2.amazonaws.com"
            },
            "Action": [
                # If the key is for encryption
                "kms:Encrypt",
                "kms:GenerateDataKey"
                # If the key is for decryption
                "kms:Decrypt"
            ],
            "Resource": "*"
        }
   ]
}
```

Autoriser les utilisateurs à supprimer des robots

Cet exemple montre un exemple de politique qui permet aux utilisateurs IAM de supprimer n'importe quel bot. La politique inclut les autorisations permettant d'effectuer cette action sur la console ou à l'aide de l' AWS API AWS CLI or.

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
          "lex:DeleteBot",
          "lex:DeleteBotLocale",
          "lex:DeleteBotAlias",
          "lex:DeleteIntent",
          "lex:DeleteSlot",
          "lex:DeleteSlottype"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:lex:Region:123412341234:bot/*",
                   "arn:aws:lex:Region:123412341234:bot-alias/*"]
    }
  ]
}
```

Permettre aux utilisateurs d'avoir une conversation avec un bot

Cet exemple montre un exemple de politique qui permet aux utilisateurs d'IAM d'avoir une conversation avec n'importe quel bot. La politique inclut les autorisations permettant d'effectuer cette action sur la console ou à l'aide de l' AWS API AWS CLI or.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
          "lex:StartConversation",
          "lex:RecognizeText",
          "lex:RecognizeUtterance",
          "lex:GetSession",
          "lex:PutSession",
          "lex:DeleteSession"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:lex:Region:123412341234:bot-alias/*"
    }
  ]
}
```

Autoriser un utilisateur spécifique à gérer les politiques basées sur les ressources

L'exemple suivant autorise un utilisateur spécifique à gérer les politiques basées sur les ressources. Il permet d'accéder à la console et à l'API aux politiques associées aux robots et aux alias de robots.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ResourcePolicyEditor",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:role/ResourcePolicyEditor"
            },
            "Action": [
                "lex:CreateResourcePolicy",
                "lex:UpdateResourcePolicy",
                "lex:DeleteResourcePolicy",
                "lex:DescribeResourcePolicy"
            ]
        }
    ]
}
```

Autoriser un utilisateur à exporter des robots et des paramètres régionaux de robots

La politique d'autorisation IAM suivante permet à un utilisateur de créer, de mettre à jour et d'exporter un bot ou des paramètres régionaux du bot.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "lex:CreateExport",
                "lex:UpdateExport",
                "lex:DescribeExport",
                "lex:DescribeBot",
                "lex:DescribeBotLocale",
                "lex:ListBotLocales",
                "lex:DescribeIntent",
                "lex:ListIntents",
                "lex:DescribeSlotType",
```

```
                "lex:ListSlotTypes",
                "lex:DescribeSlot",
                "lex:ListSlots",
                "lex:DescribeCustomVocabulary"
            ],
            "Effect": "Allow",
            "Resource": ["arn:aws:lex:Region:123456789012:bot/*"]
        }
    ]
}
```

Permettre à un utilisateur d'exporter un vocabulaire personnalisé

La politique d'autorisation IAM suivante permet à un utilisateur d'exporter un vocabulaire personnalisé à partir des paramètres régionaux d'un bot.

```
{"Version": "2012-10-17",
    "Statement": [
        {"Action": [
                "lex:CreateExport",
                "lex:UpdateExport",
                "lex:DescribeExport",
                "lex:DescribeCustomVocabulary"
            ],
            "Effect": "Allow",
            "Resource": ["arn:aws:lex:Region:123456789012:bot/*"]
        }
    ]
}
```

Autoriser un utilisateur à importer des robots et des paramètres régionaux de robots

La politique d'autorisation IAM suivante permet à un utilisateur d'importer un bot ou des paramètres régionaux du bot et de vérifier le statut d'une importation.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "lex:CreateUploadUrl",
                "lex:StartImport",
```

```
                "lex:DescribeImport",
                "lex:CreateBot",
                "lex:UpdateBot",
                "lex:DeleteBot",
                "lex:CreateBotLocale",
                "lex:UpdateBotLocale",
                "lex:DeleteBotLocale",
                "lex:CreateIntent",
                "lex:UpdateIntent",
                "lex:DeleteIntent",
                "lex:CreateSlotType",
                "lex:UpdateSlotType",
                "lex:DeleteSlotType",
                "lex:CreateSlot",
                "lex:UpdateSlot",
                "lex:DeleteSlot",
                "lex:CreateCustomVocabulary",
                "lex:UpdateCustomVocabulary",
                "lex:DeleteCustomVocabulary",
                "iam:PassRole",
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:lex:Region:123456789012:bot/*",
                "arn:aws:lex:Region:123456789012:bot-alias/*"
            ]
        }
    ]
}
```

Permettre à un utilisateur d'importer un vocabulaire personnalisé

La politique d'autorisation IAM suivante permet à un utilisateur d'importer un vocabulaire personnalisé dans les paramètres régionaux d'un bot.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "lex:CreateUploadUrl",
                "lex:StartImport",
                "lex:DescribeImport",
```

```
                "lex:CreateCustomVocabulary",
                "lex:UpdateCustomVocabulary",
                "lex:DeleteCustomVocabulary"

            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:lex:Region:123456789012:bot/*"
            ]
        }
    ]
}
```

Autoriser un utilisateur à migrer un bot d'Amazon Lex vers Amazon Lex V2

La politique d'autorisation IAM suivante permet à un utilisateur de commencer à migrer un bot d'Amazon Lex vers Amazon Lex V2.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "startMigration",
            "Effect": "Allow",
            "Action": "lex:StartMigration",
            "Resource": "arn:aws:lex:>Region<:>123456789012<:bot:*"
        },
        {
            "Sid": "passRole",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::>123456789012<:role/>v2 bot role<"
        },
        {
            "Sid": "allowOperations",
            "Effect": "Allow",
            "Action": [
                "lex:CreateBot",
                "lex:CreateIntent",
                "lex:UpdateSlot",
                "lex:DescribeBotLocale",
                "lex:UpdateBotAlias",
                "lex:CreateSlotType",
```

```
                "lex:DeleteBotLocale",
                "lex:DescribeBot",
                "lex:UpdateBotLocale",
                "lex:CreateSlot",
                "lex:DeleteSlot",
                "lex:UpdateBot",
                "lex:DeleteSlotType",
                "lex:DescribeBotAlias",
                "lex:CreateBotLocale",
                "lex:DeleteIntent",
                "lex:StartImport",
                "lex:UpdateSlotType",
                "lex:UpdateIntent",
                "lex:DescribeImport",
                "lex:CreateCustomVocabulary",
                "lex:UpdateCustomVocabulary",
                "lex:DeleteCustomvocabulary",
                "lex:DescribeCustomVocabulary",
                "lex:DescribeCustomVocabularyMetadata"
            ],
            "Resource": [
                "arn:aws:lex:>Region<:>123456789012<:bot/*",
                "arn:aws:lex:>Region<:>123456789012<:bot-alias/*/*"
            ]
        },
        {
            "Sid": "showBots",
            "Effect": "Allow",
            "Action": [
                "lex:CreateUploadUrl",
                "lex:ListBots"
            ],
            "Resource": "*"
        }
    ]
}
```

Autorisation accordée aux utilisateurs pour afficher leurs propres autorisations

Cet exemple montre comment créer une politique qui permet aux utilisateurs IAM d'afficher les politiques en ligne et gérées attachées à leur identité d'utilisateur. Cette politique inclut les autorisations permettant d'effectuer cette action sur la console ou par programmation à l'aide de l'API AWS CLI or AWS .

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Permettre à un utilisateur de créer un flux de conversation avec le générateur de conversation visuel dans Amazon Lex V2

La politique d'autorisation IAM suivante permet à un utilisateur de dessiner le flux de conversation à l'aide du générateur de conversation visuel dans Amazon Lex V2.

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {"Action": [
                "lex:UpdateIntent ",
                "lex:DescribeIntent "
            ],
            "Effect": "Allow",
            "Resource": ["arn:aws:lex:Region:123456789012:bot/*"]
        }
    ]
}
```

Autoriser les utilisateurs à créer et à afficher des répliques de robots, mais pas à les supprimer

Vous pouvez associer les autorisations suivantes à un rôle IAM pour lui permettre de créer et d'afficher uniquement des répliques de robots. En omettant`lex:DeleteBotReplica`, vous empêchez le rôle de supprimer les répliques de robots. Pour plus d'informations, consultez Autorisations pour répliquer des robots et gérer les répliques de robots.

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "lex:CreateBotReplica",
            "lex:DescribeBotReplica",
            "lex:ListBotReplica",
            "lex:ListBotVersionReplicas",
            "lex:ListBotAliasReplicas",
         ],
         "Resource": [
             "arn:aws:lex:*:*:bot/*",
             "arn:aws:lex:*:*:bot-alias/*"
         ]
     },
      {
         "Effect": "Allow",
         "Action": [
             "iam:GetRole"
         ],
         "Resource": [
```

```
            "arn:aws:iam::*:role/aws-service-role/replication.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Replication*"
         ]
      },
      {
         "Effect": "Allow",
         "Action": [
             "iam:CreateServiceLinkedRole",
         ],
         "Resource": [
             "arn:aws:iam::*:role/aws-service-role/replication.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Replication*"
         ],
         "Condition": {
             "StringEquals": {
                 "iam:AWSServiceName": "lexv2.amazonaws.com"
             }
         }
      }
    ]
}
```

Exemples de politiques basées sur les ressources pour Amazon Lex V2

Une politique basée sur les ressources est attachée à une ressource, telle qu'un bot ou un alias de bot. Avec une politique basée sur les ressources, vous pouvez spécifier qui a accès à la ressource et les actions qu'il peut effectuer sur celle-ci. Par exemple, vous pouvez ajouter des politiques basées sur les ressources qui permettent à un utilisateur de modifier un bot spécifique ou d'utiliser des opérations d'exécution sur un alias de bot spécifique.

Lorsque vous utilisez une politique basée sur les ressources, vous pouvez autoriser d'autres AWS services à accéder aux ressources de votre compte. Par exemple, vous pouvez autoriser Amazon Connect à accéder à un robot Amazon Lex.

Pour savoir comment créer un bot ou un alias de bot, consultezConstruire des robots.

Rubriques

- Utiliser la console pour définir une politique basée sur les ressources
- Utiliser l'API pour définir une politique basée sur les ressources

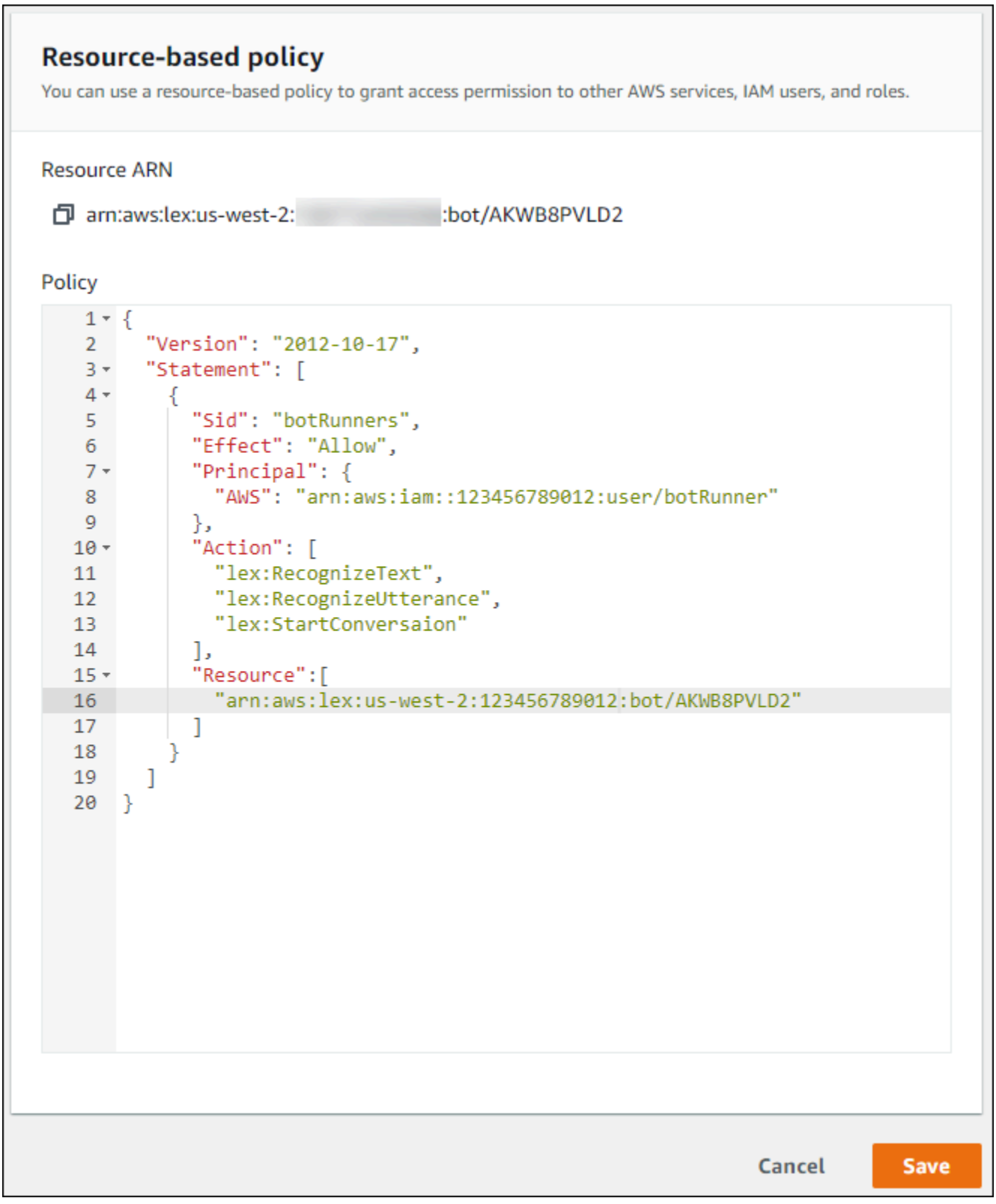
- Autoriser un rôle IAM à mettre à jour un bot et à répertorier les alias du bot
- Autoriser un utilisateur à avoir une conversation avec un bot
- Autoriser un AWS service à utiliser un bot Amazon Lex V2 spécifique

Utiliser la console pour définir une politique basée sur les ressources

Vous pouvez utiliser la console Amazon Lex pour gérer les politiques basées sur les ressources pour vos robots et alias de robots. Vous entrez la structure JSON d'une politique et la console l'associe à la ressource. Si une politique est déjà associée à une ressource, vous pouvez utiliser la console pour afficher et modifier la politique.

Lorsque vous enregistrez une politique dans l'éditeur de stratégie, la console vérifie la syntaxe de la stratégie. Si la politique contient des erreurs, telles qu'un utilisateur inexistant ou une action non prise en charge par la ressource, elle renvoie une erreur et n'enregistre pas la politique.

Ce qui suit montre l'éditeur de stratégie basé sur les ressources pour un bot dans la console. L'éditeur de politique pour un alias de bot est similaire.


Resource-based policy
You can use a resource-based policy to grant access permission to other AWS services, IAM users, and roles.
Resource ARN
arn:aws:lex:us-west-2: :bot/AKWB8PVLD2
Policy
{
"Version": "2012-10-17",
"Statement": [
{
"Sid": "botRunners",
"Effect": "Allow",
"Principal": {
"AWS": "arn:aws:iam::123456789012:user/botRunner"
},
"Action": [
"lex:RecognizeText",
"lex:RecognizeUtterance",
"lex:StartConversaion"
],
"Resource":[
"arn:aws:lex:us-west-2:123456789012:bot/AKWB8PVLD2"
]
}
]
}
Cancel
Save

Pour ouvrir l'éditeur de politiques pour un bot

1. Connectez-vous à la console Amazon Lex AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/lex/.
2. Dans la liste des robots, choisissez le bot dont vous souhaitez modifier la politique.
3. Dans la section Stratégie basée sur les ressources, choisissez Modifier.

Pour ouvrir l'éditeur de politiques pour un alias de bot

1. Connectez-vous à la console Amazon Lex AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/lex/.
2. Dans la liste des robots, choisissez le bot qui contient l'alias que vous souhaitez modifier.
3. Dans le menu de gauche, choisissez Alias, puis choisissez l'alias à modifier.
4. Dans la section Stratégie basée sur les ressources, choisissez Modifier.

Utiliser l'API pour définir une politique basée sur les ressources

Vous pouvez utiliser les opérations d'API pour gérer les politiques basées sur les ressources pour vos robots et alias de robots. Des opérations permettent de créer, de mettre à jour et de supprimer des politiques.

CreateResourcePolicy

Ajoute une nouvelle politique de ressources avec les déclarations de politique spécifiées à un bot ou à un alias de bot.

CreateResourcePolicyStatement

Ajoute une nouvelle déclaration de politique de ressources à un bot ou à un alias de bot.

DeleteResourcePolicy

Supprime une politique de ressources d'un bot ou d'un alias de bot.

DeleteResourcePolicyStatement

Supprime une déclaration de politique de ressources d'un bot ou d'un alias de bot.

DescribeResourcePolicy

Obtient une politique de ressources et la révision de la politique.

UpdateResourcePolicy

Remplace la politique de ressources existante pour un bot ou un alias de bot par une nouvelle.

Exemples

Java

L'exemple suivant montre comment utiliser les opérations de stratégie basées sur les ressources pour gérer une stratégie basée sur les ressources.

```
        /*
         * Create a new policy for the specified bot alias
         * that allows a role to invoke lex:UpdateBotAlias on it.
         * The created policy will have revision id 1.
         */

        CreateResourcePolicyRequest createPolicyRequest =
                CreateResourcePolicyRequest.builder()
                        .resourceArn("arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID")
                        .policy("{\"Version\": \"2012-10-17\",\"Statement
\": [{\"Sid\": \"BotAliasEditor\",\"Effect\": \"Allow\",\"Principal\":
 {\"AWS\": \"arn:aws:iam::123456789012:role/BotAliasEditor\"},\"Action\":
 [\"lex:UpdateBotAlias\"],\"Resource\":[\"arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID\"]]}")

        lexmodelsv2Client.createResourcePolicy(createPolicyRequest);

        /*
         * Overwrite the policy for the specified bot alias with a new policy.
         * Since no expectedRevisionId is provided, this request overwrites the
 current revision.
         * After this update, the revision id for the policy is 2.
         */
        UpdateResourcePolicyRequest updatePolicyRequest =
        UpdateResourcePolicyRequest.builder()
                        .resourceArn("arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID")
                        .policy("{\"Version\": \"2012-10-17\",\"Statement
\": [{\"Sid\": \"BotAliasEditor\",\"Effect\": \"Deny\",\"Principal\":
 {\"AWS\": \"arn:aws:iam::123456789012:role/BotAliasEditor\"},\"Action\":
```

```
 [\"lex:UpdateBotAlias\"],\"Resource\":[\"arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID\"]]}")

        lexmodelsv2Client.updateResourcePolicy(updatePolicyRequest);

        /*
         * Creates a statement in an existing policy for the specified bot alias
         * that allows a role to invoke lex:RecognizeText on it.
         * This request expects to update revision 2 of the policy. The request will
 fail
         * if the current revision of the policy is no longer revision 2.
         * After this request, the revision id for this policy will be 3.
         */

        CreateResourcePolicyStatementRequest createStatementRequest =
                CreateResourcePolicyStatementRequest.builder()
                        .resourceArn("arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID")
                        .effect("Allow")

 .principal(Principal.builder().arn("arn:aws:iam::123456789012:role/
BotRunner").build())
                        .action("lex:RecognizeText")
                        .statementId("BotRunnerStatement")
                        .expectedRevisionId(2)
                        .build();

        lexmodelsv2Client.createResourcePolicyStatement(createStatementRequest);

        /*
         * Deletes a statement from an existing policy for the specified bot alias
 by statementId.
         * Since no expectedRevisionId is supplied, the request will remove the
 statement from
         * the current revision of the policy for the bot alias.
         * After this request, the revision id for this policy will be 4.
         */
        DeleteResourcePolicyRequest deleteStatementRequest =
                DeleteResourcePolicyRequest.builder()
                        .resourceArn("arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID")
                        .statementId("BotRunnerStatement")
                        .build();
```

```
        lexmodelsv2Client.deleteResourcePolicy(deleteStatementRequest);

        /*
         * Describe the current policy for the specified bot alias
         * It always returns the current revision.
         */
        DescribeResourcePolicyRequest describePolicyRequest =
                DescribeResourcePolicyRequest.builder()
                        .resourceArn("arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID")
                        .build();

        lexmodelsv2Client.describeResourcePolicy(describePolicyRequest);


        /*
         * Delete the current policy for the specified bot alias
         * This request expects to delete revision 3 of the policy. Since the
 revision id for
         * this policy is already at 4, this request will fail.
         */
        DeleteResourcePolicyRequest deletePolicyRequest =
                DeleteResourcePolicyRequest.builder()
                        .resourceArn("arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID")
                        .expectedRevisionId(3);
                        .build();

        lexmodelsv2Client.deleteResourcePolicy(deletePolicyRequest);
```

Autoriser un rôle IAM à mettre à jour un bot et à répertorier les alias du bot

L'exemple suivant accorde des autorisations à un rôle IAM spécifique pour appeler les opérations d'API de création de modèles Amazon Lex V2 afin de modifier un bot existant. L'utilisateur peut répertorier les alias d'un bot et le mettre à jour, mais ne peut pas supprimer le bot ou les alias du bot.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "botBuilders",
            "Effect": "Allow",
```

```
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:role/BotBuilder"
            },
            "Action": [
                "lex:ListBotAliases",
                "lex:UpdateBot"
            ],
            "Resource": [
                "arn:aws:lex:Region:123456789012:bot/MYBOT"
            ]
        }
    ]
}
```

Autoriser un utilisateur à avoir une conversation avec un bot

L'exemple suivant autorise un utilisateur spécifique à appeler les opérations de l'API d'exécution Amazon Lex V2 sur un alias unique d'un bot.

L'utilisateur se voit spécifiquement refuser l'autorisation de mettre à jour ou de supprimer l'alias du bot.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "botRunners",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/botRunner"
            },
            "Action": [
                "lex:RecognizeText",
                "lex:RecognizeUtterance",
                "lex:StartConversation",
                "lex:DeleteSession",
                "lex:GetSession",
                "lex:PutSession"
            ],
            "Resource": [
                "arn:aws:lex:Region:123456789012:bot-alias/MYBOT/MYBOTALIAS"
            ]
        },
```

```
        {
            "Sid": "botRunners",
            "Effect": "Deny",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/botRunner"
            },
            "Action": [
                "lex:UpdateBotAlias",
                "lex:DeleteBotAlias"
            ],
            "Resource": [
                "arn:aws:lex:Region:123456789012:bot-alias/MYBOT/MYBOTALIAS"
            ]
        }
    ]
}
```

Autoriser un AWS service à utiliser un bot Amazon Lex V2 spécifique

L'exemple suivant autorise Amazon Connect à appeler les AWS Lambda opérations de l'API d'exécution Amazon Lex V2.

Le bloc de condition est obligatoire pour les principaux de service et doit utiliser les clés de contexte globales `AWS:SourceAccount` et`AWS:SourceArn`.

`AWS:SourceAccount`Il s'agit de l'ID de compte qui appelle le bot Amazon Lex V2.

`AWS:SourceArn`Il s'agit de l'ARN de ressource de l'instance de service Amazon Connect ou de la fonction Lambda d'où provient l'appel à l'alias du bot Amazon Lex V2.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "connect-bot-alias",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "connect.amazonaws.com"
                ]
            },
            "Action": [
                "lex:RecognizeText",
```

```
                "lex:StartConversation"
            ],
            "Resource": [
                "arn:aws:lex:Region:123456789012:bot-alias/MYBOT/MYBOTALIAS"
            ],
            "Condition": {
                "StringEquals": {
                    "AWS:SourceAccount": "123456789012"
                },
                "ArnEquals": {
                    "AWS:SourceArn":
 "arn:aws:connect:Region:123456789012:instance/instance-id"
                }
            }
        },
        {
            "Sid": "lambda-function",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "lambda.amazonaws.com"
                ]
            },
            "Action": [
                "lex:RecognizeText",
                "lex:StartConversation"
            ],
            "Resource": [
                "arn:aws:lex:Region:123456789012:bot-alias/MYBOT/MYBOTALIAS"
            ],
            "Condition": {
                "StringEquals": {
                    "AWS:SourceAccount": "123456789012"
                },
                "ArnEquals": {
                    "AWS:SourceArn":
 "arn:aws:lambda:Region:123456789012:function/function-name"
                }
            }
        }
    ]
}
```

AWS politiques gérées pour Amazon Lex V2

Une politique AWS gérée est une politique autonome créée et administrée par AWS. AWS les politiques gérées sont conçues pour fournir des autorisations pour de nombreux cas d'utilisation courants afin que vous puissiez commencer à attribuer des autorisations aux utilisateurs, aux groupes et aux rôles.

N'oubliez pas que les politiques AWS gérées peuvent ne pas accorder d'autorisations de moindre privilège pour vos cas d'utilisation spécifiques, car elles sont accessibles à tous les AWS clients. Nous vous recommandons de réduire encore les autorisations en définissant des politiques gérées par le client qui sont propres à vos cas d'utilisation.

Vous ne pouvez pas modifier les autorisations définies dans les politiques AWS gérées. Si les autorisations définies dans une politique AWS gérée sont AWS mises à jour, la mise à jour affecte toutes les identités principales (utilisateurs, groupes et rôles) auxquelles la politique est attachée. AWS est le plus susceptible de mettre à jour une politique AWS gérée lorsqu'une nouvelle Service AWS est lancée ou lorsque de nouvelles opérations d'API sont disponibles pour les services existants.

Pour plus d'informations, consultez la section Politiques gérées par AWS dans le Guide de l'utilisateur IAM.

Politique gérée par AWS : AmazonLexReadOnly

Vous pouvez associer la politique `AmazonLexReadOnly` à vos identités IAM.

Cette politique accorde des autorisations en lecture seule qui permettent aux utilisateurs de visualiser toutes les actions dans le service de création de modèles Amazon Lex V2 et Amazon Lex.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes :

- `lex`— Accès en lecture seule aux ressources Amazon Lex V2 et Amazon Lex dans le service de modélisme.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "lex:GetBot",
                "lex:GetBotAlias",
                "lex:GetBotAliases",
                "lex:GetBots",
                "lex:GetBotChannelAssociation",
                "lex:GetBotChannelAssociations",
                "lex:GetBotVersions",
                "lex:GetBuiltinIntent",
                "lex:GetBuiltinIntents",
                "lex:GetBuiltinSlotTypes",
                "lex:GetIntent",
                "lex:GetIntents",
                "lex:GetIntentVersions",
                "lex:GetSlotType",
                "lex:GetSlotTypes",
                "lex:GetSlotTypeVersions",
                "lex:GetUtterancesView",
                "lex:DescribeBot",
                "lex:DescribeBotAlias",
                "lex:DescribeBotChannel",
                "lex:DescribeBotLocale",
                "lex:DescribeBotRecommendation",
                "lex:DescribeBotVersion",
                "lex:DescribeCustomVocabulary",
                "lex:DescribeCustomVocabularyMetadata",
                "lex:DescribeExport",
                "lex:DescribeImport",
                "lex:DescribeIntent",
                "lex:DescribeResourcePolicy",
                "lex:DescribeSlot",
                "lex:DescribeSlotType",
                "lex:ListBotRecommendations",
                "lex:ListBots",
                "lex:ListBotLocales",
                "lex:ListBotAliases",
                "lex:ListBotChannels",
                "lex:ListBotVersions",
```

```
                "lex:ListBuiltInIntents",
                "lex:ListBuiltInSlotTypes",
                "lex:ListCustomVocabularyItems",
                "lex:ListExports",
                "lex:ListImports",
                "lex:ListIntents",
                "lex:ListRecommendedIntents",
                "lex:ListSlots",
                "lex:ListSlotTypes",
                "lex:ListTagsForResource",
                "lex:SearchAssociatedTranscripts"
            ],
            "Resource": "*"
        }
    ]
}
```

Politique gérée par AWS : AmazonLexRunBotsOnly

Vous pouvez associer la politique `AmazonLexRunBotsOnly` à vos identités IAM.

Cette politique accorde des autorisations en lecture seule qui permettent d'accéder aux robots conversationnels Amazon Lex V2 et Amazon Lex.

Détails de l’autorisation

Cette politique inclut les autorisations suivantes :

- `lex`— Accès en lecture seule à toutes les actions dans Amazon Lex V2 et Amazon Lex Runtime.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "lex:PostContent",
                "lex:PostText",
                "lex:PutSession",
                "lex:GetSession",
                "lex:DeleteSession",
                "lex:RecognizeText",
                "lex:RecognizeUtterance",
```

```
                "lex:StartConversation"
            ],
            "Resource": "*"
        }
    ]
}
```

Politique gérée par AWS : AmazonLexFullAccess

Vous pouvez associer la politique `AmazonLexFullAccess` à vos identités IAM.

Cette politique accorde des autorisations administratives qui autorisent l'utilisateur à créer, lire, mettre à jour et supprimer des ressources Amazon Lex V2 et Amazon Lex, ainsi qu'à exécuter les robots conversationnels Amazon Lex V2 et Amazon Lex.

Détails de l’autorisation

Cette politique inclut les autorisations suivantes :

- `lex`— Permet aux principaux d'accéder en lecture et en écriture à toutes les actions des services de création et d'exécution de modèles Amazon Lex V2 et Amazon Lex.
- `cloudwatch`— Permet aux directeurs de consulter les CloudWatch métriques et les alarmes d'Amazon.
- `iam`— Permet aux principaux de créer et de supprimer des rôles liés à un service, de transmettre des rôles et d'associer et de détacher des politiques à un rôle. Les autorisations sont limitées à « lex.amazonaws.com » pour les opérations Amazon Lex et à « lexv2.amazonaws.com » pour les opérations Amazon Lex V2.
- `kendra`— Permet aux principaux de répertorier les index Amazon Kendra.
- `kms`— Permet aux principaux de décrire les AWS KMS clés et les alias.
- `lambda`— Permet aux principaux de répertorier les AWS Lambda fonctions et de gérer les autorisations associées à n'importe quelle fonction Lambda.
- `polly`— Permet aux directeurs de décrire les voix d'Amazon Polly et de synthétiser le discours.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AmazonLexFullAccessStatement1",
```

```
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:DescribeAlarmsForMetric",
                "kms:DescribeKey",
                "kms:ListAliases",
                "lambda:GetPolicy",
                "lambda:ListFunctions",
                "lambda:ListAliases",
                "lambda:ListVersionsByFunction"
                "lex:*",
                "polly:DescribeVoices",
                "polly:SynthesizeSpeech",
                "kendra:ListIndices",
                "iam:ListRoles",
                "s3:ListAllMyBuckets",
                "logs:DescribeLogGroups",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "AmazonLexFullAccessStatement2",
            "Effect": "Allow",
            "Action": [
                "bedrock:ListFoundationModels"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "bedrock:InvokeModel"
            ],
            "Resource": "arn:aws:bedrock:*::foundation-model/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:AddPermission",
                "lambda:RemovePermission"
```

```
            ],
            "Resource": "arn:aws:lambda:*:*:function:AmazonLex*",
            "Condition": {
                "StringEquals": {
                    "lambda:Principal": "lex.amazonaws.com"
                }
            }
        },
        {
            "Sid": "AmazonLexFullAccessStatement3",
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "iam:GetRolePolicy"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots",
                "arn:aws:iam::*:role/aws-service-role/channels.lex.amazonaws.com/
AWSServiceRoleForLexChannels",
                "arn:aws:iam::*:role/aws-service-role/lexv2.amazonaws.com/
AWSServiceRoleForLexV2Bots*",
                "arn:aws:iam::*:role/aws-service-role/channels.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Channels*",
                "arn:aws:iam::*:role/aws-service-role/replication.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Replication*"
            ]
        },
        {
            "Sid": "AmazonLexFullAccessStatement4",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "lex.amazonaws.com"
                }
            }
        },
```

```
        {
            "Sid": "AmazonLexFullAccessStatement5",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/channels.lex.amazonaws.com/
AWSServiceRoleForLexChannels"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "channels.lex.amazonaws.com"
                }
            }
        },
        {
            "Sid": "AmazonLexFullAccessStatement6",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/lexv2.amazonaws.com/
AWSServiceRoleForLexV2Bots*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "lexv2.amazonaws.com"
                }
            }
        },
        {
            "Sid": "AmazonLexFullAccessStatement7",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/channels.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Channels*"
            ],
            "Condition": {
                "StringEquals": {
```

```
                    "iam:AWSServiceName": "channels.lexv2.amazonaws.com"
                }
            }
        },
        {
            "Sid": "AmazonLexFullAccessStatement8",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/replication.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Replication*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "lexv2.amazonaws.com"
                }
            }
        },
        {
            "Sid": "AmazonLexFullAccessStatement9",
            "Effect": "Allow",
            "Action": [
                "iam:DeleteServiceLinkedRole",
                "iam:GetServiceLinkedRoleDeletionStatus"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots",
                "arn:aws:iam::*:role/aws-service-role/channels.lex.amazonaws.com/
AWSServiceRoleForLexChannels",
                "arn:aws:iam::*:role/aws-service-role/lexv2.amazonaws.com/
AWSServiceRoleForLexV2Bots*",
                "arn:aws:iam::*:role/aws-service-role/channels.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Channels*",
                "arn:aws:iam::*:role/aws-service-role/replication.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Replication*"
            ]
        },
        {
            "Sid": "AmazonLexFullAccessStatement10",
            "Effect": "Allow",
            "Action": [
```

```
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "lex.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Sid": "AmazonLexFullAccessStatement11",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/lexv2.amazonaws.com/
AWSServiceRoleForLexV2Bots*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "lexv2.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Sid": "AmazonLexFullAccessStatement12",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/channels.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Channels*"
            ],
            "Condition": {
                "StringEquals": {
```

```
                    "iam:PassedToService": [
                        "channels.lexv2.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Sid": "AmazonLexFullAccessStatement13",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/replication.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Replication*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "lexv2.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

Politique gérée par AWS : AmazonLexReplicationPolicy

Vous ne pouvez pas joindre de `AmazonLexReplicationPolicy` à vos entités IAM. Cette politique est associée à un rôle lié à un service qui permet à Amazon Lex V2 d'effectuer des actions en votre nom. Pour plus d’informations, consultez Utilisation de rôles liés à un service pour Amazon Lex V2.

Cette politique accorde des autorisations administratives qui permettent à Amazon Lex V2 de répliquer les AWS ressources entre les régions en votre nom. Vous pouvez associer cette politique pour permettre à un rôle de répliquer facilement des ressources, notamment des robots, des paramètres régionaux, des versions, des alias, des intentions, des types d'emplacements, des emplacements et des vocabulaires personnalisés.

Détails de l’autorisation

Cette politique inclut les autorisations suivantes.

- `lex`— Permet aux directeurs de répliquer les ressources dans d'autres régions.
- `iam`— Permet aux directeurs de transmettre des rôles depuis IAM. Cela est nécessaire pour qu'Amazon Lex V2 soit autorisé à répliquer des ressources dans d'autres régions.

```
{
    "Version": "2012-10-17",
    "Statement": [
 {
  "Sid": "ReplicationPolicyStatement1",
  "Effect": "Allow",
  "Action": [
   "lex:BuildBotLocale",
   "lex:ListBotLocales",
   "lex:CreateBotAlias",
   "lex:UpdateBotAlias",
   "lex:DeleteBotAlias",
   "lex:DescribeBotAlias",
   "lex:CreateBotVersion",
   "lex:DeleteBotVersion",
   "lex:DescribeBotVersion",
   "lex:CreateExport",
   "lex:DescribeBot",
   "lex:UpdateExport",
   "lex:DescribeExport",
   "lex:DescribeBotLocale",
   "lex:DescribeIntent",
   "lex:ListIntents",
   "lex:DescribeSlotType",
   "lex:ListSlotTypes",
   "lex:DescribeSlot",
   "lex:ListSlots",
   "lex:DescribeCustomVocabulary",
   "lex:StartImport",
   "lex:DescribeImport",
   "lex:CreateBot",
   "lex:UpdateBot",
   "lex:DeleteBot",
   "lex:CreateBotLocale",
```

```
    "lex:UpdateBotLocale",
    "lex:DeleteBotLocale",
    "lex:CreateIntent",
    "lex:UpdateIntent",
    "lex:DeleteIntent",
    "lex:CreateSlotType",
    "lex:UpdateSlotType",
    "lex:DeleteSlotType",
    "lex:CreateSlot",
    "lex:UpdateSlot",
    "lex:DeleteSlot",
    "lex:CreateCustomVocabulary",
    "lex:UpdateCustomVocabulary",
    "lex:DeleteCustomVocabulary",
    "lex:DeleteBotChannel",
    "lex:DeleteResourcePolicy"
   ],
   "Resource": [
    "arn:aws:lex:*:*:bot/*",
    "arn:aws:lex:*:*:bot-alias/*"
   ]
  },
  {
   "Sid": "ReplicationPolicyStatement2",
   "Effect": "Allow",
   "Action": [
    "lex:CreateUploadUrl",
    "lex:ListBots"
   ],
   "Resource": "*"
  },
  {
   "Sid": "ReplicationPolicyStatement3",
   "Effect": "Allow",
   "Action": [
    "iam:PassRole"
   ],
   "Resource": "*",
   "Condition": {
    "StringEquals": {
     "iam:PassedToService": "lexv2.amazonaws.com"
    }
   }
  }
```

```
    ]
}
```

Amazon Lex V2 met à jour les politiques AWS gérées

Consultez les informations relatives aux mises à jour des politiques AWS gérées pour Amazon Lex V2 depuis que ce service a commencé à suivre ces modifications. Pour recevoir des alertes automatiques concernant les modifications apportées à cette page, abonnez-vous au flux RSS sur la Historique du document pour Amazon Lex V2 page Amazon Lex V2.

Modification	Description	Date
AmazonLexFullAccess – Mise à jour d'une stratégie existante	Amazon Lex V2 a ajouté de nouvelles autorisations pour permettre la réplication des ressources du bot vers d'autres régions.	31 janvier 2024
AmazonLexReplicationPolicy – Nouvelle politique	Amazon Lex V2 a ajouté une nouvelle politique permettant la réplication des ressources des robots vers d'autres régions.	31 janvier 2024
AmazonLexReadOnly - mise à jour d'une politique existante	Amazon Lex V2 a ajouté de nouvelles autorisations pour permettre un accès en lecture seule pour répertorier des éléments de vocabulaire personnalisés.	29 novembre 2022
AmazonLexReadOnly - mise à jour d'une politique existante	Amazon Lex V2 a ajouté de nouvelles autorisations pour permettre l'accès en lecture seule aux informati	16 novembre 2021

Modification	Description	Date
	ons relatives aux vocabulaires personnalisés.	
AmazonLexFullAccess - mise à jour d'une politique existante	Amazon Lex V2 a ajouté de nouvelles autorisations pour permettre un accès en lecture seule aux opération s du service de modélisme Amazon Lex V2.	18 août 2021
AmazonLexReadOnly - mise à jour d'une politique existante	Amazon Lex V2 a ajouté de nouvelles autorisations pour permettre un accès en lecture seule aux opération s du concepteur de Chatbot automatisé Amazon Lex V2.	1er décembre 2021
AmazonLexFullAccess - mise à jour d'une politique existante	Amazon Lex V2 a ajouté de nouvelles autorisations pour permettre un accès en lecture seule aux opération s du service de modélisme Amazon Lex V2.	18 août 2021
AmazonLexReadOnly - mise à jour d'une politique existante	Amazon Lex V2 a ajouté de nouvelles autorisations pour permettre un accès en lecture seule aux opération s du service de modélisme Amazon Lex V2.	18 août 2021

Modification	Description	Date
AmazonLexRunBotsOnly - mise à jour d'une politique existante	Amazon Lex V2 a ajouté de nouvelles autorisations pour permettre l'accès en lecture seule aux opérations du service d'exécution Amazon Lex V2.	18 août 2021
Amazon Lex V2 a commencé à suivre les modifications	Amazon Lex V2 a commencé à suivre les modifications apportées AWS à ses politique s gérées.	18 août 2021

Utilisation de rôles liés à un service pour Amazon Lex V2

Amazon Lex V2 utilise des AWS Identity and Access Management rôles liés à un service (IAM). Un rôle lié à un service est un type unique de rôle IAM directement lié à Amazon Lex V2. Les rôles liés à un service sont prédéfinis par Amazon Lex V2 et incluent toutes les autorisations requises par le service pour appeler d'autres AWS services en votre nom.

Un rôle lié à un service facilite la configuration d'Amazon Lex V2, car vous n'avez pas à ajouter manuellement les autorisations nécessaires. Amazon Lex V2 définit les autorisations associées à ses rôles liés aux services et, sauf indication contraire, seul Amazon Lex V2 peut assumer ses rôles. Les autorisations définies comprennent la politique d'approbation et la politique d'autorisation. De plus, cette politique d'autorisation ne peut pas être attachée à une autre entité IAM.

Pour de plus amples informations sur les autres services qui prennent en charge les rôles liés à un service, veuillez consulter Services AWS qui fonctionnent avec IAM et rechercher les services qui ont Yes (Oui) dans la colonne Service-Linked Role (Rôle lié à un service). Sélectionnez un Yes (Oui) avec un lien permettant de consulter la documentation du rôle lié à un service, pour ce service.

Vous devez configurer les autorisations de manière à permettre à une entité IAM (comme un utilisateur, un groupe ou un rôle) de créer, modifier ou supprimer un rôle lié à un service. Pour plus d'informations, consultez Service-Linked Role Permissions (autorisations du rôle lié à un service) dans le IAM User Guide (guide de l'utilisateur IAM).

Vous ne pouvez supprimer un rôle lié à un service qu'après avoir d'abord supprimé les ressources associées. Cela protège vos ressources Amazon Lex V2, car vous ne pouvez pas supprimer par inadvertance les autorisations d'accès aux ressources.

Rubriques

Création d'un rôle lié à un service pour Amazon Lex V2

Il n'est pas nécessaire de créer manuellement un rôle lié à un service, car Amazon Lex V2 crée le rôle lié au service pour vous lorsque vous effectuez l'action appropriée (voir Autorisations de rôle liées à un service pour Amazon Lex V2 pour plus d'informations) dans l'API AWS Management Console, AWS CLI ou. AWS

Si vous supprimez ce rôle lié au service, puis que vous devez en créer un de nouveau, vous pouvez utiliser le même processus pour créer un nouveau rôle dans votre compte.

Modification d'un rôle lié à un service pour Amazon Lex V2

Amazon Lex V2 ne vous permet pas de modifier les rôles liés à un service. Une fois que vous avez créé un rôle lié à un service, vous ne pouvez pas changer le nom du rôle, car plusieurs entités peuvent faire référence au rôle. Vous pouvez toutefois modifier la description d'un rôle à l'aide d'IAM. Pour plus d’informations, consultez Modification d’un rôle lié à un service dans le Guide de l’utilisateur IAM.

Suppression d'un rôle lié à un service pour Amazon Lex V2

Si vous n'avez plus besoin d'utiliser une fonction ou un service qui nécessite un rôle lié à un service, nous vous recommandons de supprimer ce rôle. De cette façon, vous n’avez aucune entité inutilisée qui n’est pas surveillée ou gérée activement. Cependant, vous devez nettoyer les ressources de votre rôle lié à un service avant de pouvoir les supprimer manuellement.

> **Note**
>
> Si le service Amazon Lex V2 utilise le rôle lorsque vous essayez de supprimer les ressources, la suppression risque d'échouer. Si cela se produit, patientez quelques minutes et réessayez.

Pour connaître les étapes à suivre pour supprimer des ressources pour des rôles spécifiques liés à un service dans Amazon Lex V2, reportez-vous à la section spécifique au rôle dans. Autorisations de rôle liées à un service pour Amazon Lex V2

Pour supprimer manuellement un rôle lié à un service à l'aide d'IAM

Après avoir supprimé les ressources associées à un rôle lié à un service, utilisez la console IAM AWS CLI, le ou l' AWS API pour supprimer le rôle. Pour plus d'informations, veuillez consulter Suppression d'un rôle lié à un service dans le Guide de l'utilisateur IAM.

Autorisations de rôle liées à un service pour Amazon Lex V2

Amazon Lex V2 utilise des rôles liés à un service avec les préfixes suivants.

Rubriques

- AWSServiceRoleForLexV2Bots_
- AWSServiceRoleForLexV2Channels_
- AWSServiceRoleForLexV2Replication

AWSServiceRoleForLexV2Bots_

Le rôle AWSServiceRoleForLexV2Bots _ donne l'autorisation de connecter votre bot aux autres services requis. Ce rôle inclut une politique de confiance permettant au service lexv2.amazonaws.com d'assumer le rôle et inclut des autorisations pour effectuer les actions suivantes.

- Utilisez Amazon Polly pour synthétiser le discours sur toutes les ressources Amazon Lex V2 prises en charge par l'action.
- Si un bot est configuré pour utiliser l'analyse des sentiments d'Amazon Comprehend, détectez le sentiment sur toutes les ressources Amazon Lex V2 prises en charge par l'action.

- Si un bot est configuré pour stocker des journaux audio dans un compartiment S3, placez les objets dans un compartiment spécifié.
- Si un bot est configuré pour stocker des journaux audio et textuels, créez un flux de journaux et placez les journaux dans un groupe de journaux spécifié.
- Si un bot est configuré pour utiliser une AWS KMS clé pour chiffrer des données, générez une clé de données spécifique.
- Si un bot est configuré pour utiliser l'`KendraSearchIntent`intention, demandez l'accès à un index Amazon Kendra spécifié.

Pour créer le rôle

Amazon Lex V2 crée un nouveau rôle AWSServiceRoleForLexV2Bots _ avec un suffixe aléatoire dans votre compte chaque fois que vous créez un bot. Amazon Lex V2 modifie le rôle lorsque vous ajoutez des fonctionnalités supplémentaires à un bot. Par exemple, si vous ajoutez l'analyse des sentiments d'Amazon Comprehend à un bot, Amazon Lex V2 ajoute l'autorisation pour l'`lex:DetectSentiment`action au rôle de service.

Pour supprimer le rôle

1. Connectez-vous à la console Amazon Lex AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/lex/.
2. Dans le volet de navigation de gauche, sélectionnez Bots et choisissez le bot dont vous souhaitez supprimer le rôle lié au service.
3. Sélectionnez n'importe quelle version du bot.
4. Le rôle d'exécution des autorisations IAM se trouve dans les détails de la version.
5. Retournez à la page Bots et cliquez sur le bouton radio situé à côté du bot à supprimer.
6. Sélectionnez Action, puis sélectionnez Supprimer.
7. Suivez les étapes décrites dans Supprimer un rôle lié à un service pour supprimer le rôle IAM.

AWSServiceRoleForLexV2Channels_

Le rôle AWSServiceRoleForLexV2Channels _ donne l'autorisation de répertorier les robots dans un compte et d'appeler les API de conversation d'un bot. Ce rôle inclut une politique de confiance permettant au service channels.lexv2.amazonaws.com d'assumer ce rôle. Si un bot est configuré pour utiliser un canal pour communiquer avec un service de messagerie, la politique d'autorisations

AWSServiceRoleForLexV2Channels _ role permet à Amazon Lex V2 d'effectuer les actions suivantes.

- Répertoriez les autorisations sur tous les robots d'un compte.
- Reconnaissez le texte, obtenez une session et attribuez des autorisations de session à un alias de bot spécifié.

Pour créer le rôle

Lorsque vous créez une intégration de canaux pour déployer un bot sur une plateforme de messagerie, Amazon Lex V2 crée un nouveau rôle lié à un service dans votre compte pour chaque canal avec un suffixe aléatoire.

Pour supprimer le rôle

1. Connectez-vous à la console Amazon Lex AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/lex/.
2. Dans le volet de navigation de gauche, sélectionnez Bots.
3. Choisissez un robot.
4. Dans le volet de navigation de gauche, choisissez Channel integrations sous Deployments.
5. Sélectionnez une chaîne dont vous souhaitez supprimer le rôle lié au service.
6. Le rôle d'exécution des autorisations IAM se trouve dans la configuration générale
7. Choisissez Supprimer, puis sélectionnez à nouveau Supprimer pour supprimer la chaîne.
8. Suivez les étapes décrites dans Supprimer un rôle lié à un service pour supprimer le rôle IAM.

AWSServiceRoleForLexV2Replication

Le AWSServiceRoleForLexV2Replication rôle donne l'autorisation de répliquer des robots dans une deuxième région. Ce rôle inclut une politique de confiance permettant au service replication.lexv2.amazonaws.com d'assumer le rôle et inclut également la politique AmazonLexReplicationPolicy AWS gérée, qui autorise les autorisations pour les actions suivantes.

- Transmettez les rôles IAM du robot au robot de réplication afin de dupliquer les autorisations appropriées pour le robot de réplication.
- Créez et gérez des robots et des ressources de robots (versions, alias, intentions, emplacements, vocabulaires personnalisés, etc.) dans d'autres régions.

Pour créer le rôle

Lorsque vous activez Global Resiliency pour un bot, Amazon Lex V2 crée le rôle AWSServiceRoleForLexV2Replication lié au service dans votre compte. Assurez-vous de disposer des autorisations appropriées pour accorder au service Amazon Lex V2 les autorisations nécessaires à la création du rôle lié au service.

Pour supprimer les ressources Amazon Lex V2 utilisées par AWSServiceRoleForLexV2Replication afin de pouvoir supprimer le rôle

1. Connectez-vous à la console Amazon Lex AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/lex/.
2. Choisissez un bot pour lequel Global Resiliency est activé.
3. Sélectionnez Global Resiliency sous Déploiement.
4. Sélectionnez Désactiver la résilience globale.
5. Répétez le processus pour tous les robots sur lesquels Global Resiliency est activé.
6. Suivez les étapes décrites dans Supprimer un rôle lié à un service pour supprimer le rôle IAM.

Régions prises en charge pour les rôles liés au service Amazon Lex V2

Amazon Lex V2 prend en charge l'utilisation de rôles liés au service dans toutes les régions où le service est disponible. Pour plus d’informations, consultez Régions et points de terminaison AWS.

Résolution des problèmes d'identité et d'accès à Amazon Lex V2

Utilisez les informations suivantes pour vous aider à diagnostiquer et à résoudre les problèmes courants que vous pouvez rencontrer lorsque vous travaillez avec Amazon Lex V2 et IAM.

Rubriques

- Je ne suis pas autorisé à effectuer une action dans Amazon Lex V2
- Je ne suis pas autorisé à effectuer iam : PassRole
- Je suis administrateur et je souhaite autoriser d'autres personnes à accéder à Amazon Lex V2
- Accorder un accès programmatique à un utilisateur
- Je souhaite autoriser des personnes extérieures à mon AWS compte à accéder à mes ressources Amazon Lex V2

Je ne suis pas autorisé à effectuer une action dans Amazon Lex V2

S'il vous AWS Management Console indique que vous n'êtes pas autorisé à effectuer une action, vous devez contacter votre administrateur pour obtenir de l'aide. Votre administrateur est la personne qui vous a fourni vos informations de connexion.

L'exemple d'erreur suivant se produit quand l'utilisateur IAM `mateojackson` tente d'utiliser la console pour afficher des informations détaillées sur une ressource *`my-example-widget`* fictive, mais ne dispose pas des autorisations `lex:`*`GetWidget`* fictives.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 lex:GetWidget on resource: my-example-widget
```

Dans ce cas, Mateo demande à son administrateur de mettre à jour ses politiques pour lui permettre d'accéder à la ressource *`my-example-widget`* à l'aide de l'action `lex:`*`GetWidget`*.

Je ne suis pas autorisé à effectuer iam : PassRole

Si vous recevez un message d'erreur indiquant que vous n'êtes pas autorisé à effectuer l'`iam:PassRole`action, vos politiques doivent être mises à jour pour vous permettre de transmettre un rôle à Amazon Lex V2.

Certains vous Services AWS permettent de transmettre un rôle existant à ce service au lieu de créer un nouveau rôle de service ou un rôle lié à un service. Pour ce faire, un utilisateur doit disposer des autorisations nécessaires pour transmettre le rôle au service.

L'exemple d'erreur suivant se produit lorsqu'un utilisateur IAM nommé `marymajor` essaie d'utiliser la console pour effectuer une action dans Amazon Lex V2. Toutefois, l'action nécessite que le service ait des autorisations accordées par un rôle de service. Mary ne dispose pas des autorisations nécessaires pour transférer le rôle au service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Dans ce cas, les stratégies de Mary doivent être mises à jour pour lui permettre d'exécuter l'action `iam:PassRole`.

Si vous avez besoin d'aide, contactez votre AWS administrateur. Votre administrateur vous a fourni vos informations d'identification de connexion.

Je suis administrateur et je souhaite autoriser d'autres personnes à accéder à Amazon Lex V2

Pour autoriser d'autres personnes à accéder à Amazon Lex V2, vous devez créer une entité IAM (utilisateur ou rôle) pour la personne ou l'application qui doit y accéder. Ils utiliseront les informations d'identification de cette entité pour accéder à AWS. Vous devez ensuite joindre une politique à l'entité qui lui accorde les autorisations appropriées dans Amazon Lex V2.

Pour démarrer immédiatement, consultez Création de votre premier groupe et utilisateur délégué IAM dans le Guide de l'utilisateur IAM.

Accorder un accès programmatique à un utilisateur

Les utilisateurs ont besoin d'un accès programmatique s'ils souhaitent interagir avec AWS l'extérieur du AWS Management Console. La manière d'accorder un accès programmatique dépend du type d'utilisateur qui y accède AWS.

Pour accorder aux utilisateurs un accès programmatique, choisissez l'une des options suivantes.

Quel utilisateur a besoin d'un accès programmatique ?	Pour	Méthode
Identité de la main-d'œuvre (Utilisateurs gérés dans IAM Identity Center)	Utilisez des informations d'identification temporaires pour signer les demandes programmatiques adressées aux AWS CLI AWS SDK ou AWS aux API.	Suivez les instructions de l'interface que vous souhaitez utiliser. • Pour le AWS CLI, voir Configuration du AWS CLI à utiliser AWS IAM Identity Center dans le guide de AWS Command Line Interface l'utilisateur. • Pour les AWS SDK, les outils et les AWS API, consultez la section Authentification IAM Identity Center dans le Guide de

Quel utilisateur a besoin d'un accès programmatique ?	Pour	Méthode
		référence AWS des SDK et des outils.
IAM	Utilisez des informations d'identification temporaires pour signer les demandes programmatiques adressées aux AWS CLI AWS SDK ou AWS aux API.	Suivez les instructions de la section Utilisation d'informations d'identification temporaires avec AWS les ressources du Guide de l'utilisateur IAM.
IAM	(Non recommandé) Utilisez des informations d'identification à long terme pour signer les AWS CLI demandes programmatiques adressées aux AWS SDK ou AWS aux API.	Suivez les instructions de l'interface que vous souhaitez utiliser. • Pour le AWS CLI, voir Authentification à l'aide des informations d'identification utilisateur IAM dans le Guide de l'AWS Command Line Interface utilisateur. • Pour les AWS SDK et les outils, voir Authentifier à l'aide d'informations d'identification à long terme dans le Guide de AWS référence des SDK et des outils. • Pour les AWS API, consultez la section Gestion des clés d'accès pour les utilisateurs IAM dans le guide de l'utilisateur IAM.

Je souhaite autoriser des personnes extérieures à mon AWS compte à accéder à mes ressources Amazon Lex V2

Vous pouvez créer un rôle que les utilisateurs provenant d'autres comptes ou les personnes extérieures à votre organisation pourront utiliser pour accéder à vos ressources. Vous pouvez spécifier qui est autorisé à assumer le rôle. Pour les services qui prennent en charge les politiques basées sur les ressources ou les listes de contrôle d'accès (ACL), vous pouvez utiliser ces politiques pour donner l'accès à vos ressources.

Pour en savoir plus, consultez les éléments suivants :

- Pour savoir si Amazon Lex V2 prend en charge ces fonctionnalités, consultezComment Amazon Lex V2 fonctionne avec IAM.
- Pour savoir comment fournir l'accès à vos ressources sur celles Comptes AWS que vous possédez, consultez la section Fournir l'accès à un utilisateur IAM dans un autre utilisateur Compte AWS que vous possédez dans le Guide de l'utilisateur IAM.
- Pour savoir comment fournir l'accès à vos ressources à des tiers Comptes AWS, consultez la section Fournir un accès à des ressources Comptes AWS détenues par des tiers dans le guide de l'utilisateur IAM.
- Pour savoir comment fournir un accès par le biais de la fédération d'identité, consultez Fournir un accès à des utilisateurs authentifiés en externe (fédération d'identité) dans le Guide de l'utilisateur IAM.
- Pour découvrir quelle est la différence entre l'utilisation des rôles et l'utilisation des politiques basées sur les ressources pour l'accès intercompte, consultez Différence entre les rôles IAM et les politiques basées sur les ressources dans le Guide de l'utilisateur IAM.

Journalisation et surveillance dans Amazon Lex V2

La surveillance joue un rôle important dans le maintien de la fiabilité, de la disponibilité et des performances d'Amazon Lex V2 et de vos autres solutions AWS. AWS fournit les outils de surveillance suivants pour surveiller Amazon Lex V2, signaler un problème et prendre des mesures automatiques le cas échéant :

- Amazon CloudWatch surveille vos AWS ressources et les applications que vous utilisez AWS en temps réel. Vous pouvez collecter et suivre les métriques, créer des tableaux de bord personnalisés, et définir des alarmes qui vous informent ou prennent des mesures lorsqu'une

métrique spécifique atteint un seuil que vous spécifiez. Par exemple, vous pouvez CloudWatch suivre l'utilisation du processeur ou d'autres indicateurs de vos instances Amazon EC2 et lancer automatiquement de nouvelles instances en cas de besoin. Pour plus d'informations, consultez le guide de CloudWatch l'utilisateur Amazon.

- AWS CloudTrailcapture les appels d'API et les événements associés effectués par ou pour le compte de votre AWS compte et envoie les fichiers journaux dans un compartiment Amazon S3 que vous spécifiez. Vous pouvez identifier les utilisateurs et les comptes appelés AWS, l'adresse IP source à partir de laquelle les appels ont été effectués et la date des appels. Pour plus d'informations, consultez le Guide de l'utilisateur AWS CloudTrail.

Validation de conformité pour Amazon Lex V2

Des auditeurs tiers évaluent la sécurité et la conformité d'Amazon Lex V2 dans le cadre de plusieurs programmes de AWS conformité. Amazon Lex V2 est un service éligible à la loi HIPAA. Il est conforme aux normes PCI, SOC et ISO.

Pour savoir si un programme Services AWS de conformité Service AWS s'inscrit dans le champ d'application de programmes de conformité spécifiques, consultez Services AWS la section de conformité et sélectionnez le programme de conformité qui vous intéresse. Pour des informations générales, voir Programmes de AWS conformité Programmes AWS de .

Vous pouvez télécharger des rapports d'audit tiers à l'aide de AWS Artifact. Pour plus d'informations, voir Téléchargement de rapports dans AWS Artifact .

Votre responsabilité en matière de conformité lors de l'utilisation Services AWS est déterminée par la sensibilité de vos données, les objectifs de conformité de votre entreprise et les lois et réglementations applicables. AWS fournit les ressources suivantes pour faciliter la mise en conformité :

- Guides de démarrage rapide sur la sécurité et la conformité : ces guides de déploiement abordent les considérations architecturales et indiquent les étapes à suivre pour déployer des environnements de base axés sur AWS la sécurité et la conformité.
- Architecture axée sur la sécurité et la conformité HIPAA sur Amazon Web Services : ce livre blanc décrit comment les entreprises peuvent créer des applications AWS conformes à la loi HIPAA.

> **Note**
>
> Tous ne Services AWS sont pas éligibles à la loi HIPAA. Pour plus d'informations, consultez le HIPAA Eligible Services Reference.

- AWS Ressources de https://aws.amazon.com/compliance/resources/ de conformité — Cette collection de classeurs et de guides peut s'appliquer à votre secteur d'activité et à votre région.
- AWS Guides de conformité destinés aux clients — Comprenez le modèle de responsabilité partagée sous l'angle de la conformité. Les guides résument les meilleures pratiques en matière de sécurisation Services AWS et décrivent les directives relatives aux contrôles de sécurité dans de nombreux cadres (notamment le National Institute of Standards and Technology (NIST), le Payment Card Industry Security Standards Council (PCI) et l'Organisation internationale de normalisation (ISO)).
- Évaluation des ressources à l'aide des règles du guide du AWS Config développeur : le AWS Config service évalue dans quelle mesure les configurations de vos ressources sont conformes aux pratiques internes, aux directives du secteur et aux réglementations.
- AWS Security Hub— Cela Service AWS fournit une vue complète de votre état de sécurité interne AWS. Security Hub utilise des contrôles de sécurité pour évaluer vos ressources AWS et vérifier votre conformité par rapport aux normes et aux bonnes pratiques du secteur de la sécurité. Pour obtenir la liste des services et des contrôles pris en charge, consultez Référence des contrôles Security Hub.
- AWS Audit Manager— Cela vous Service AWS permet d'auditer en permanence votre AWS utilisation afin de simplifier la gestion des risques et la conformité aux réglementations et aux normes du secteur.

Résilience dans Amazon Lex V2

L'infrastructure AWS mondiale est construite autour des AWS régions et des zones de disponibilité. AWS Les régions fournissent plusieurs zones de disponibilité physiquement séparées et isolées, connectées par un réseau à faible latence, à haut débit et hautement redondant. Avec les zones de disponibilité, vous pouvez concevoir et exploiter des applications et des bases de données qui basculent automatiquement d'une zone à l'autre sans interruption. Les zones de disponibilité sont davantage disponibles, tolérantes aux pannes et ont une plus grande capacité de mise à l'échelle que les infrastructures traditionnelles à un ou plusieurs centres de données.

Pour plus d'informations sur AWS les régions et les zones de disponibilité, consultez la section Infrastructure AWS mondiale.

Outre l'infrastructure AWS mondiale, Amazon Lex V2 propose plusieurs fonctionnalités pour répondre à vos besoins en matière de résilience et de sauvegarde des données.

> **Note**
>
> Pour plus d'informations sur la résilience globale dans Amazon Lex V2, qui vous permet de créer un bot répliqué dans une deuxième région par paires prédéterminées, consultez Global Resiliency.

Sécurité de l'infrastructure dans Amazon Lex V2

En tant que service géré, Amazon Lex V2 est protégé par les procédures de sécurité du réseau AWS mondial décrites dans le livre blanc Amazon Web Services : présentation des processus de sécurité.

Vous utilisez des appels d'API AWS publiés pour accéder à Amazon Lex V2 via le réseau. Les clients doivent supporter le protocole TLS (Sécurité de la couche transport) 1.0 ou une version ultérieure. Nous recommandons TLS 1.2 ou version ultérieure. Les clients doivent aussi prendre en charge les suites de chiffrement PFS (Perfect Forward Secrecy) comme Ephemeral Diffie-Hellman (DHE) ou Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). La plupart des systèmes modernes tels que Java 7 et les versions ultérieures prennent en charge ces modes.

En outre, les demandes doivent être signées à l'aide d'un ID de clé d'accès et d'une clé d'accès secrète associée à un principal IAM. Vous pouvez également utiliser AWS Security Token Service (AWS STS) pour générer des informations d'identification de sécurité temporaires et signer les demandes.

Amazon Lex V2 et points de terminaison VPC d'interface ()AWS PrivateLink

Vous pouvez établir une connexion privée entre votre VPC et Amazon Lex V2 en créant un point de terminaison VPC d'interface. Les points de terminaison de l'interface sont alimentés par AWS PrivateLinkune technologie qui vous permet d'accéder en privé aux API Amazon Lex V2 sans passerelle Internet, appareil NAT, connexion VPN ou connexion AWS Direct Connect. Les instances

de votre VPC n'ont pas besoin d'adresses IP publiques pour communiquer avec les API Amazon Lex V2. Le trafic entre votre VPC et Amazon Lex V2 ne quitte pas le réseau Amazon.

Chaque point de terminaison d'interface est représenté par une ou plusieurs interfaces réseau Elastic dans vos sous-réseaux.

Pour plus d'informations, consultez la section Interface VPC endpoints (AWS PrivateLink) dans le guide de l'utilisateur Amazon VPC.

Considérations relatives aux points de terminaison VPC Amazon Lex V2

Avant de configurer un point de terminaison VPC d'interface pour Amazon Lex V2, assurez-vous de consulter les propriétés et les limites du point de terminaison d'interface dans le guide de l'utilisateur Amazon VPC.

Amazon Lex V2 permet d'appeler toutes ses actions d'API depuis votre VPC.

Création d'un point de terminaison VPC d'interface pour Amazon Lex V2

Vous pouvez créer un point de terminaison VPC pour le service Amazon Lex V2 à l'aide de la console Amazon VPC ou du (). AWS Command Line Interface AWS CLI Pour plus d'informations, consultez Création d'un point de terminaison d'interface dans le Guide de l'utilisateur Amazon VPC.

Créez un point de terminaison VPC pour Amazon Lex V2 en utilisant le nom de service suivant :

- com.amazonaws.*region*.models-v2-lex
- com.amazonaws.*region*.runtime-v2-lex

Si vous activez le DNS privé pour le point de terminaison, vous pouvez envoyer des demandes d'API à Amazon Lex V2 en utilisant son nom DNS par défaut pour la région, par exemple,`runtime-v2-lex.us-east-1.amazonaws.com`.

Pour plus d'informations, consultez Accès à un service via un point de terminaison d'interface dans le Guide de l'utilisateur Amazon VPC.

Création d'une politique de point de terminaison VPC pour Amazon Lex V2

Vous pouvez associer une politique de point de terminaison à votre point de terminaison VPC qui contrôle l'accès à Amazon Lex V2. La politique spécifie les informations suivantes :

- Le principal qui peut exécuter des actions.
- Les actions qui peuvent être effectuées.
- Les ressources sur lesquelles les actions peuvent être exécutées.

Pour plus d'informations, consultez Contrôle de l'accès aux services avec points de terminaison d'un VPC dans le Guide de l'utilisateur Amazon VPC.

Exemple : politique de point de terminaison VPC pour les actions Amazon Lex V2

Voici un exemple de politique de point de terminaison pour Amazon Lex V2. Lorsqu'elle est attachée à un point de terminaison, cette politique accorde l'accès aux actions Amazon Lex V2 répertoriées à tous les principaux sur toutes les ressources.

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "lex:RecognizeText",
            "lex:RecognizeUtterance",
            "lex:StartConversation",
            "lex:DeleteSession",
            "lex:GetSession",
            "lex:DeleteSession"
         ],
         "Resource":"*"
      }
   ]
}
```

Consignes et bonnes pratiques

Reportez-vous aux directives et bonnes pratiques suivantes pour optimiser le comportement de votre bot et ses interactions avec les clients.

Signature des requêtes

Toutes les demandes de création de modèles et d'exécution d'Amazon Lex V2 figurant dans la référence d'API utilisent la signature V4 pour authentifier les demandes. Pour plus d'informations sur l'authentification des demandes, consultez la section Processus de signature de la version 4 du. Références générales AWS

Protection des informations confidentielles

Les opérations RecognizeTextde l'API d'exécution RecognizeUtteranceprennent un ID de session comme paramètre obligatoire. Les développeurs peuvent lui attribuer n'importe quelle valeur conformes aux contraintes décrites dans l'API. Nous vous recommandons de ne pas utiliser ce paramètre pour envoyer des informations confidentielles, telles que les identifiants d'utilisateur, les e-mails ou les numéros de sécurité sociale. Cet identifiant est principalement utilisé pour identifier de manière unique une conversation avec un bot.

Capture des valeurs d'emplacement à partir des déclarations de l'utilisateur

Amazon Lex V2 utilise les valeurs d'énumération que vous fournissez dans la définition d'un type d'emplacement pour entraîner ses modèles d'apprentissage automatique. Supposons que vous définissiez une intention appelée `GetPredictionIntent` à l'aide de l'exemple d'énoncé suivant :

```
"Tell me the prediction for {sign}"
```

où {sign} est un emplacement dont le type `ZodiacSign` personnalisé comporte 12 valeurs d'énumération : `Aries` à. `Pisces` Supposons maintenant que l'utilisateur dise « Dites-moi la prédiction pour la Terre » :

- Amazon Lex V2 en déduit que « terre » est une ZodiacSign valeur si vous effectuez l'une des actions suivantes :
  - Définissez le `valueSelectionStrategy` champ pour `ORIGINAL_VALUE` utiliser l'CreateSlotTypeopération
  - Sélectionnez Développer les valeurs dans la console

- Amazon Lex V2 ne reconnaît pas la valeur « earth » si vous limitez la reconnaissance aux valeurs que vous avez définies pour le type d'emplacement en effectuant l'une des actions suivantes :
  - Définissez le `valueSelectionStrategy` champ pour `TOP_RESOLUTION` utiliser l'`CreateSlotType`opération
  - Sélectionnez Restreindre aux valeurs des emplacements et aux synonymes dans la console

Lorsque vous définissez des synonymes pour des valeurs d'intervalle, ils sont reconnus comme étant identiques à une valeur de créneau. Toutefois, la valeur de l'emplacement est renvoyée à la place du synonyme.

Étant donné qu'Amazon Lex V2 transmet cette valeur à votre application cliente ou à la fonction Lambda, vous devez vérifier que les valeurs des créneaux sont des valeurs valides avant de les utiliser dans votre activité de traitement des commandes.

Lorsqu'Amazon Lex V2 appelle une fonction Lambda ou renvoie le résultat d'une interaction vocale avec votre client, la validité des valeurs des slots n'est pas garantie. Dans des interactions textuelles, la casse des valeurs d'option correspond au texte saisi ou à la valeur d'option, en fonction de la valeur du champ `valueResolutionStrategy`.

Acronymes utilisés dans les valeurs des créneaux

Lorsque vous définissez des valeurs d'emplacement contenant des acronymes, utilisez les modèles suivants :

- Lettres majuscules séparées par des points (D.V.D.)
- Lettres majuscules séparées par des espaces (D V D)

Emplacements intégrés pour la date et l'heure

Les types Amazon. Date de créneaux Amazon Time intégrés capturent les dates et les heures (absolues et relatives). Les dates et heures relatives sont résolues à l'heure et à la date auxquelles Amazon Lex V2 reçoit la demande et dans la région où la demande est traitée.

Pour le type de créneau `AMAZON.Time` intégré, si l'utilisateur ne précise pas qu'une heure se situe avant ou après midi, l'heure est ambiguë. Dans ce cas, Amazon Lex V2 invite à nouveau l'utilisateur. Nous vous recommandons d'utiliser des invites qui impliquent une heure absolue. Par exemple, utilisez une invite, telles que « When do you want your pizza delivered? You can say 6 PM or 6 in the evening ».

Éviter toute ambiguïté dans les données d'entraînement de votre bot

Le fait de fournir des données d'entraînement confuses dans votre bot réduit la capacité d'Amazon Lex V2 à comprendre les entrées des utilisateurs. Supposons que votre bot ait deux intentions (`OrderPizza`et`OrderDrink`) et que vous incluiez « Je veux commander » comme exemple d'énoncé. Lorsque vous créez votre bot, Amazon Lex V2 n'est pas en mesure de faire correspondre cet énoncé à une intention spécifique. Par conséquent, lorsqu'un utilisateur saisit cet énoncé lors de l'exécution, Amazon Lex V2 ne peut pas définir une intention avec un degré de confiance élevé.

Si vous avez deux intentions avec le même exemple d'énoncé, utilisez des contextes de saisie pour aider Amazon Lex V2 à faire la distinction entre les deux intentions lors de l'exécution. Pour plus d'informations, consultez la section Définition du contexte d'intention.

Utilisation de l'alias TSTALIASID

- L'alias TSTALIASID de votre bot pointe vers la version préliminaire et ne doit être utilisé que pour des tests manuels. Amazon Lex limite le nombre de demandes d'exécution que vous pouvez envoyer à l'alias TSTALIASID du bot.
- Lorsque vous mettez à jour la version préliminaire du bot, Amazon Lex arrête toutes les conversations en cours pour toute application cliente utilisant l'alias TSTALIASID du bot. En règle générale, vous ne devez pas utiliser l'alias TSTALIASID d'un bot en production car la version préliminaire peut être mise à jour. Vous devez publier une version et un alias et les utiliser à la place.
- Lorsque vous mettez à jour un alias, Amazon Lex prend quelques minutes pour récupérer les modifications. Lorsque vous modifiez la version préliminaire du bot, la modification est immédiatement prise en compte par l'alias TSTALIASID.

Quotas

Les quotas de service, également appelés limites, sont le nombre maximum de ressources de service autorisées pour votre AWS compte. Pour plus d'informations, consultez la section Quotas de service AWS dans la référence AWS générale.

Certains quotas de service peuvent être réglés ou augmentés. Reportez-vous à la colonne Réglable des tableaux suivants pour savoir si un quota peut être ajusté et à la colonne Self-service pour savoir si vous pouvez demander un ajustement de quota via la console des quotas de service. Contactez-nous AWS Support pour augmenter un quota ajustable, mais pas par le biais du libre-service. L'augmentation d'un quota de service peut prendre quelques jours. Si vous augmentez votre quota dans le cadre d'un projet plus vaste, veillez à ajouter ce temps à votre plan.

> **Note**
>
> Les limites de caractères sont calculées comme le nombre d'unités de code Unicode. Dans la plupart des cas, un caractère Unicode équivaut à une unité de code Unicode. Certains caractères spéciaux peuvent être supérieurs à une unité et les nombres peuvent différer selon les encodages. Pour plus d'informations sur le calcul de la longueur de chaîne, consultez cette documentation.

Quotas de durée de construction

Les quotas maximaux suivants sont appliqués lorsque vous créez un bot.

Description	Par défaut	Ajustable	Libre-service
Bots par AWS compte	100	Oui	Oui
Associations de canaux de bots par AWS compte	5 000	Non	N/A
Bots par réseau de robots	5	Non	N/A

Description	Par défaut	Ajustable	Libre-service
Réseaux de bots par bot	25	Non	N/A
Versions par bot	100	Non	N/A
Intentions par région dans chaque bot	• 1 000 en en-AU, en-GB et en-US • 250 dans toutes les autres régions	Oui	Non
Machines à sous par région dans chaque bot	• 4 000 en en-AU, en-GB et en-US • 2 000 dans toutes les autres régions	Non	N/A
Types de machines à sous personnalisés par bot local	• 250 en en-AU, en-GB et en-US • 100 dans toutes les autres régions	Non	N/A
Valeurs de type d'emplacement personnalisées et synonymes par région dans chaque bot	50 000	Non	N/A
Nombre total de caractères dans les exemples d'énoncés par langue locale dans chaque bot	• 2 000 000 en en-AU, en-GB et en-US • 200 000 dans toutes les autres régions	Non	N/A

Description	Par défaut	Ajustable	Libre-service
Associations de canaux par alias de bot	10	Non	N/A
Nombre de machines à sous par intention	100	Non	N/A
Exemples d'énoncés par intention	1 500	Oui	Oui
Nombre de caractères par extrait d'énoncé	500	Non	N/A
Longueur de la réponse du texte	4 000	Non	N/A
Exemples d'énoncés par emplacement	10	Oui	Oui
Nombre de caractères par intervalle d'extrait d'énoncé	500	Non	N/A
Invites par emplacement	30	Non	N/A
Valeurs et synonymes par type de slot personnalisé	10 000	Non	N/A
Nombre de caractères par valeur de type de slot personnalisé	500	Non	N/A
Caractères du nom d'une association de chaînes	100	Non	N/A

Description	Par défaut	Ajustable	Libre-service
Nombre de tâches d'analyse simultanées effectuées par Automated Chatbot Designer sur tous les bots de votre compte, par région	10	Non	N/A
Taille du fichier XML de type slot de grammaire personnalisé	100 Ko	Non	N/A

Quotas d’exécution

Les quotas maximaux suivants sont appliqués au moment de l'exécution.

Description	Par défaut	Ajustable	Libre-service
Taille du texte de saisie pour Recognize Textet Recognize Utterance	1024 caractères	Non	N/A
Longueur d'entrée vocale pour le `Recognize Utterance` fonctionnement	15 secondes	Oui	Non
Taille des `Recognize Utterance` en-têtes	16 Ko	Non	N/A

Description	Par défaut	Ajustable	Libre-service
Taille des en-têtes de demande et de session combinés pour `RecognizeUtterance`	12 KO	Non	N/A
Nombre maximum de conversations simultanées en mode texte pour `RecognizeText` `RecognizeUtterance`, ou `StartConversation` pour TestBotAlias	2	Non	N/A
Nombre maximum de conversations simultanées en mode texte pour `RecognizeText` ou `StartConversation` pour `RecognizeUtterance` d'autres alias	50	Oui	Non
Nombre maximum de conversations simultanées en mode vocal pour `RecognizeUtterance` TestBotAlias	2	Non	N/A

Description	Par défaut	Ajustable	Libre-service
Nombre maximum de conversations simultanées en mode vocal `RecognizeUtterance` pour les autres alias	125	Oui	Non
Nombre maximum de conversations simultanées en mode vocal pour `StartConversation` TestBotAlias	2	Non	N/A
Nombre maximum de conversations simultanées en mode vocal `StartConversation` pour les autres alias	200	Oui	Non
Nombre maximal d'opérations de gestion de session simultanées (`PutSession` ,`GetSession` , ou`DeleteSession`) lors de l'utilisation du TestBotAlias	2	Non	N/A

Description	Par défaut	Ajustable	Libre-service
Nombre maximal d'opérations de gestion de session simultanées (`PutSession` ,`GetSession` , ou`DeleteSession`) lors de l'utilisation d'autres alias	50	Oui	Non
Taille d'entrée maximale pour une fonction Lambda	12 KO	Non	N/A
Taille de sortie maximale d'une fonction Lambda	25 KO	Non	N/A
Taille maximale des attributs de session dans la sortie de la fonction Lambda (après codage en base 64)	12 KO	Non	N/A
Délai d'expiration maximal d'une fonction Lambda	30 secondes	Oui	Non

Guide de migration Amazon Lex V1 vers V2

La console et les API Amazon Lex V2 facilitent la création et la gestion de robots. Utilisez ce guide pour en savoir plus sur les améliorations apportées à l'API Amazon Lex V2 lors de la migration de robots.

Vous faites migrer un bot à l'aide de la console ou de l'API Amazon Lex. Pour plus d'informations, consultez la section Migration d'un bot dans le guide du développeur Amazon Lex.

Présentation d'Amazon Lex V2

Plusieurs langues peuvent être ajoutées à un robot afin que vous puissiez les gérer comme une ressource unique. Une architecture d'informations simplifiée vous permet de gérer efficacement les versions de vos robots. Des fonctionnalités telles que le « flux de conversation », l'enregistrement partiel de la configuration du bot et le téléchargement groupé d'énoncés vous offrent une plus grande flexibilité.

Plusieurs langues dans un robot

Vous pouvez ajouter plusieurs langues à l'aide de l'API Amazon Lex V2. Vous ajoutez, modifiez et créez chaque langue indépendamment. Les ressources telles que les types de créneaux sont limitées au niveau de la langue. Vous pouvez passer rapidement d'une langue à l'autre pour comparer et affiner les conversations. Vous pouvez utiliser un tableau de bord dans la console pour consulter les énoncés dans toutes les langues afin d'accélérer les analyses et les itérations. Un opérateur de robot peut gérer les autorisations et les opérations de journalisation pour toutes les langues avec une seule configuration de robot. Vous devez fournir une langue comme paramètre d'exécution pour converser avec un robot Amazon Lex V2. Pour plus d'informations, veuillez consulter Langues et paramètres régionaux pris en charge par Amazon Lex V2.

Architecture d'information simplifiée

L'API Amazon Lex V2 suit une architecture d'informations (IA) simplifiée avec des types d'intention et d'emplacements limités à une langue. Vous effectuez la version au niveau du bot afin que les ressources telles que les intentions et les types d'emplacements ne soient pas versionnées individuellement. Par défaut, un bot est créé avec une version Draft modifiable et utilisée pour tester les modifications. Vous pouvez créer des instantanés numérotés à partir de la version préliminaire. Vous choisissez les langues à inclure dans une version. Toutes les ressources du bot (langues,

intentions, types d'emplacements) sont archivées dans le cadre de la création d'une version du bot. Pour plus d'informations, veuillez consulter Versions.

Productivité de constructeur améliorée

Vous disposez d'outils et de fonctionnalités de productivité supplémentaires pour les constructeurs qui vous offrent plus de flexibilité et de contrôle sur le processus de conception de votre robot.

Enregistrer une configuration partielle

L'API Amazon Lex V2 vous permet d'enregistrer des modifications partielles pendant le développement. Par exemple, vous pouvez enregistrer un emplacement qui fait référence à un type d'emplacement supprimé. Cette flexibilité vous permet de sauvegarder votre travail et d'y revenir plus tard. Vous pouvez résoudre ces modifications avant de créer le bot. Dans Amazon Lex V2, la sauvegarde partielle peut être appliquée aux emplacements, aux versions et aux alias.

Renommer les ressources

Avec Amazon Lex V2, vous pouvez renommer une ressource après sa création. Utilisez un nom de ressource pour associer des métadonnées conviviales à chaque ressource. L'API Amazon Lex V2 attribue à chaque ressource un identifiant de ressource unique à 10 caractères. Toutes les ressources ont un nom de ressource. Vous pouvez renommer les ressources sur le site.

- Robot
- Intention
- Type d'option
- Slot
- Alias

Vous pouvez utiliser des ID de ressource pour lire et modifier. Si vous utilisez l'AWS Command Line InterfaceAPI Amazon Lex V2 pour gérer Amazon Lex V2, des ID de ressource sont requis pour certaines commandes.

Gestion simplifiée des fonctions Lambda

Dans l'API Amazon Lex V2, vous définissez une fonction Lambda par langue au lieu d'une fonction pour chaque intention. La fonction Lambda est configurée dans l'alias de la langue et est utilisée à la fois pour la boîte de dialogue et le crochet de code de distribution. Vous pouvez toujours choisir

d'activer ou de désactiver la boîte de dialogue et les crochets de code d'exécution indépendamment pour chaque intention. Pour plus d'informations, veuillez consulter Activation d'une logique personnalisée avec des AWS Lambda fonctions.

Réglages granulaires

L'API Amazon Lex V2 déplace le seuil de confiance pour la classification des voix et des intentions du bot vers le périmètre linguistique. L'indicateur d'analyse des sentiments passe de la portée du bot à la portée de l'alias. Les paramètres d'expiration de session et de confidentialité dans l'étendue du bot, ainsi que les journaux de conversation dans l'étendue de l'alias, restent inchangés.

Intention de repli par défaut

L'API Amazon Lex V2 ajoute une intention de secours par défaut lorsque vous créez une langue. Utilisez-le pour configurer la gestion des erreurs pour votre bot au lieu de lancer des invites de gestion des erreurs spécifiques.

Mise à jour optimisée des variables de session

Avec l'API Amazon Lex V2, vous pouvez mettre à jour l'état de la session directement à l'aide RecognizeUtterancedes opérations RecognizeTextet, sans dépendre des API de session.

Création Amazon Lex V2AWS CloudFormation

Amazon Lex V2AWS CloudFormationAWS Vous créez un modèle qui décrit toutes lesAWS ressources que vous souhaitez utiliser, etAWS CloudFormation configure ces Amazon Lex V2 V2 V2 V2 V2 V2 V2 V2

Lorsque vous utilisezAWS CloudFormation, vous pouvez réutiliser votre modèle pour configurer vos ressources Amazon Lex V2 Décrivez vos ressources une seule fois, puis allouez-les autant de fois que vous le souhaitez dans plusieurs Comptes AWS et régions.

Amazon Lex V2AWS CloudFormation

Pour allouer et configurer des ressources pour Amazon Lex V2AWS CloudFormation Les modèles sont des fichiers texte formatés en JSON ou YAML. Ces modèles décrivent les ressources que vous souhaitez allouer dans vos piles AWS CloudFormation. Si JSON ou YAML ne vous est pas familier, vous pouvez utiliser AWS CloudFormation Designer pour vous aider à démarrer avec des modèles AWS CloudFormation. Pour plus d'informations, consultez Qu'est-ce que AWS CloudFormation Designer ? dans le AWS CloudFormationGuide de l'utilisateur.

Amazon Lex V2 prend en charge la création des ressources suivantes dansAWS CloudFormation :

- AWS::Lex::Bot
- AWS::Lex::BotAlias
- AWS::Lex::BotVersion
- AWS::Lex::ResourcePolicy

Pour de plus amples LeLex V2AWS CloudFormation

En savoir plus sur AWS CloudFormation

Pour en savoir plus sur AWS CloudFormation, consultez les ressources suivantes :

- AWS CloudFormation
- AWS CloudFormationguide de l'utilisateur
- Référence d'API AWS CloudFormation
- AWS CloudFormationGuide de l'utilisateur de l'interface de ligne de commande

Historique du document pour Amazon Lex V2

- Dernière mise à jour de la documentation : 22 mars 2024

Le tableau suivant décrit les modifications importantes apportées à chaque version d'Amazon Lex V2. Pour recevoir les notifications de mise à jour de cette documentation, abonnez-vous à un flux RSS.

Modification	Description	Date
Expansion de région	Amazon Lex V2 est désormais disponible en AWS GovCloud (ouest des États-Unis) (us-gov-west-1).	22 mars 2024
Mise à jour de la politique AWS gérée	Amazon Lex V2 a ajouté de nouvelles autorisations à la politique AmazonLex ReplicationPolicygérée afin de permettre la mise à jour des ressources de bots répliquées vers d'autres régions.	7 mars 2024
Nouvelle fonction	Vous pouvez utiliser la fonction fn.length () pour déterminer la longueur d'une valeur de chaîne dans Amazon Lex V2. Pour plus d'informations, consultez Branchement conditionnel - Fonctions.	4 mars 2024
Mise à jour de la fonctionnalité	Le slot intégré QnA pour les fonctionnalités d'IA générativ e dans Amazon Lex V2 est désormais GA. Pour plus d'informations, voir Optimiser la création et les performan	28 février 2024

	ces des robots grâce à l'IA générative.	
Mise à jour de la politique AWS gérée	Amazon Lex V2 a ajouté de nouvelles autorisations à la politique AmazonLex ReplicationPolicygérée afin de permettre la mise à jour des ressources de bots répliquées vers d'autres régions.	28 février 2024
Mise à jour de la politique AWS gérée	Amazon Lex V2 a ajouté de nouvelles autorisations à la politique AmazonLexFullAcces sgérée afin de permettre la réplication des ressources du bot vers d'autres régions.	8 février 2024
Nouvelle politique gérée	Amazon Lex V2 a ajouté une politique gérée fournissant des autorisations pour répliquer les ressources des robots dans d'autres régions. Pour plus d'informations, consultez AmazonLexReplicationPolicy.	8 février 2024
Nouvelle fonction	Vous pouvez utiliser la résilience globale pour répliquer votre bot dans une deuxième région AWS dans Amazon Lex V2. Pour plus d'informations, consultez Résilience globale.	8 février 2024

Nouvelle fonction	Vous pouvez désormais tirer parti des fonctionnalités d'IA générative d'Amazon Lex V2. Pour plus d'informations, voir Optimiser la création et les performances des robots grâce à l'IA générative.	29 novembre 2023
Nouvelle fonction	Amazon Lex V2 peut désormais utiliser la journalisation sélective pour capturer du texte et/ou du son au niveau de l'intention ou du créneau. Pour plus d'informations, consultez la section Journalisation sélective.	8 novembre 2023
Nouvelle fonction	Amazon Lex V2 peut désormais utiliser un emplacement intégré pour déterminer les réponses Oui/Non/Peut-être/Je ne sais pas en utilisant `AMAZON.Confirmation` Pour plus d'informations, consultez la section Types d'emplacements intégrés.	17 août 2023
Nouvelle fonction	Vous pouvez consulter les indicateurs de performance des intentions et des créneaux, ainsi que d'autres indicateurs conversationnels à l'aide du tableau de bord d'analyse. Pour plus d'informations, consultez Analyse.	18 juillet 2023

Nouvelle fonction	Vous pouvez améliorer la précision et le succès de votre bot grâce au Test Workbench . Pour plus d'informations, consultez Test Workbench.	6 juin 2023
Nouvelle fonction	Vous pouvez désormais créer des robots à partir d'un modèle pour certains secteurs commerciaux populaires. Pour plus d'informations, consultez la section Modèles de robots.	23 février 2023
Nouvelle fonction	Vous pouvez désormais combiner plusieurs robots au sein d'un réseau de robots afin de créer une expérienc e client intégrée. Pour plus d'informations, consultez la section Réseau de robots.	9 février 2023
Nouvelle fonction	Amazon Lex V2 prend désormais en charge les langues arabe du Golfe (Émirats arabes unis), cantonais (Hong Kong), finnois (Finlande), norvégien (Norvège), polonais (Pologne) et suédois (Suède). Pour plus d'informations, consultez Langues et paramètres régionaux pris en charge par Amazon Lex V2.	6 décembre 2022

Mise à jour de la politique AWS gérée	Amazon Lex V2 a ajouté de nouvelles autorisations à la politique AmazonLex ReadOnlygérée afin de permettre l'affichage d'élément s de vocabulaire personnal isés.	29 novembre 2022
Nouvelle fonction	Amazon Lex V2 peut afficher une représentation alternative pour une phrase ou un mot en utilisant la console ou les API pour personnaliser la sortie vocale en texte. Pour plus d'informations, voir Création d'un vocabulaire personnal isé pour la reconnaissance vocale.	7 novembre 2022
Nouvelle fonction	Amazon Lex V2 peut ajouter un attribut de poids à un élément d'article qui représent era le degré d'amplification de la phrase lors de la reconnais sance vocale. Pour plus d'informations, consultez la section Poids grammaticaux.	28 octobre 2022

Nouvelle fonction	Amazon Lex V2 peut être utilisé pour capturer des entrées sous forme libre provenant de l'utilisateur final, composées de mots ou de caractères à l'aide de`AMAZON.FreeFormInput` . Pour plus d'informations, consultez la section Types d'emplacements intégrés.	19 octobre 2022
Nouvelle fonction	Amazon Lex V2 peut afficher une représentation alternative pour une phrase ou un mot dans la sortie finale entre le discours et le texte. Pour plus d'informations, voir Création d'un vocabulaire personnalisé pour la reconnaissance vocale.	19 octobre 2022
Nouvelle fonction	Amazon Lex V2 prend désormais en charge les langues hindi (Inde) et néerlandais (Pays-Bas). Pour plus d'informations, consultez Langues et paramètres régionaux pris en charge par Amazon Lex V2.	14 octobre 2022

Nouvelle fonction	Une mise à jour a été apportée à la façon dont Amazon Lex V2 gère les entrées des utilisateurs. Vous pouvez désormais choisir si Amazon Lex V2 accepte les entrées texte, audio ou DTMF à tout moment du flux de conversation. Pour plus d'informations, consultez la section Attributs configurables.	22 septembre 2022
Nouvelle fonction	Une mise à jour a été apportée à la façon dont Amazon Lex V2 gère les flux de conversation. Visual Conversation Builder est un générateur de conversation par glisser-déposer qui permet de concevoir et de visualiser facilement des parcours de conversation. Pour plus d'informations, consultez Visual Conversation Builder.	14 septembre 2022

Nouvelle fonction	Une mise à jour a été apportée à la façon dont Amazon Lex V2 crée des emplacements complexes. Vous pouvez désormais créer des sous-emplacements complexes au sein des emplacements afin de gérer les intentions dans le cadre d'une conception de conversation complexe. Pour plus d'informations, consultez la section Création d'emplacements composites.	9 septembre 2022
Nouvelle fonction	Une mise à jour a été apportée à la façon dont Amazon Lex V2 gère le flux des chemins de conversation avec vos utilisateurs. Vous pouvez désormais créer des parcours de conversation complexes en ordonnant l'étape suivante de la conversation. Pour plus d'informations, consultez la section Création de chemins de conversation.	17 août 2022

Nouvelle fonction	Une mise à jour a été apportée à la façon dont Amazon Lex V2 gère le flux des conversations avec vos utilisateurs. Vous pouvez désormais créer des conversations complexes en ordonnant les instructions. Pour plus d'informations, consultez la section Configuration des invites.	5 juillet 2022
Nouvelle fonction	Une mise à jour a été apportée à la façon dont Amazon Lex V2 gère le flux des conversations avec vos utilisateurs. Vous pouvez désormais créer des conversations complexes à l'aide de conditions. Pour plus d'informations, voir Comprendre les nouveaux flux de conversation.	3 mai 2022
Nouvelle fonction	Ajout d'exemples de grammaires industrielles pour le type de slot grammatical intégré. Pour plus d'informations, voir Grammaires sectorielles.	22 mars 2022
Nouvelle fonction	Ajout de documentation sur l'intégration d'Amazon Lex V2 au SDK Amazon Chime. Pour plus d'informations, consultez le SDK Amazon Chime.	18 mars 2022

Nouvelle fonction	Amazon Lex V2 fournit désormais des scores de confiance pour les transcriptions vocales. Utilisez le score pour déterminer la bonne réponse de l'utilisateur. Pour plus d'informations, consultez la section Utilisation des scores de confiance de la transcription vocale.	27 janvier 2022
Nouvelle fonction	Vous pouvez désormais ajouter des indices contextuels et dynamiques aux machines à sous pour améliorer la précision de votre bot. Pour plus d'informations, consultez la section Utilisation d'indices pour améliorer la précision.	13 janvier 2022
Nouvelle fonction	Amazon Lex V2 ajoute la prise en charge des vocabulaires personnalisés afin d'améliorer la reconnaissance vocale pour les entrées audio. Pour plus d'informations, voir Création d'un vocabulaire personnalisé pour améliorer la reconnaissance vocale.	12 janvier 2022
Nouvelle fonction	Amazon Lex V2 est désormais compatible AWS PrivateLink. Pour plus d'informations, consultez la section Points de terminaison VPC (AWS). PrivateLink	7 janvier 2022

Nouvelle fonction	Amazon Lex V2 prend désormais en charge la langue catalane (Espagne). Pour plus d'informations, consultez Langues et paramètres régionaux pris en charge par Amazon Lex V2.	3 janvier 2022
Nouvelle fonction	Vous pouvez désormais créer des types de machines à sous en utilisant votre propre grammaire personnalisée. Pour plus d'informations, consultez la section Utilisation d'un type de slot grammatical personnalisé.	20 décembre 2021
Nouvelle fonction	AWS CloudFormation prend désormais en charge Amazon Lex V2. Pour plus d'informations, consultez les AWS CloudFormation ressources.	20 décembre 2021
Nouvelle fonction	Amazon Lex V2 prend désormais en charge les langues portugaise (Brésil), portugaise (Portugal) et mandarin (RPC). Pour plus d'informations, consultez Langues et paramètres régionaux pris en charge par Amazon Lex V2.	16 décembre 2021

Nouvelle fonction	Amazon Lex V2 fournit désormais un aperçu du concepteur de chatbot automatisé pour vous aider à commencer à créer un chatbot à partir des transcriptions des centres d'appels. Pour plus d'informations, consultez Utilisation du concepteur de Chatbot automatisé (version préliminaire).	1er décembre 2021
Nouvelle fonction	Vous pouvez désormais utiliser spell-by-letter des spell-by-word styles pour saisir des valeurs d'emplacement qu'Amazon Lex V2 a du mal à comprendre. Pour plus d'informations, consultez la section Utilisation de styles orthographiques pour capturer les valeurs des créneaux.	19 novembre 2021
Nouvelle fonction	Vous pouvez désormais utiliser les voix neuronales de synthèse vocale (NTTS) d'Amazon Polly pour les conversations audio avec vos utilisateurs. Pour plus d'informations, consultez Voices in Amazon Polly.	19 novembre 2021

Nouvelle fonction	Amazon Lex V2 prend désormais en charge la langue anglaise (Afrique du Sud). Pour plus d'informations, consultez Langues et paramètres régionaux pris en charge par Amazon Lex V2.	9 novembre 2021
Nouvelle fonction	Amazon Lex V2 prend désormais en charge la langue allemande (Autriche). Pour plus d'informations, consultez Langues et paramètres régionaux pris en charge par Amazon Lex V2.	5 novembre 2021
Nouvelle fonction	Vous pouvez désormais envoyer aux utilisateurs des messages de mise à jour qui s'affichent au début d'une fonction d'exécution et périodiquement pendant son exécution. Vous pouvez également créer des messages qui informent l'utilisateur de l'état d'exécution lorsque la fonction est terminée. Pour plus d'informations, consultez la section Configuration des mises à jour de l'état d'avancement des expéditions.	7 octobre 2021

Expansion de région	Amazon Lex V2 est désormais disponible en Afrique (Cape Town) (af-south-1) et en Asie-Pacifique (Séoul) (ap-north east-2).	22 septembre 2021
Nouvelle fonction	Vous pouvez désormais consulter les statistiques des énoncés que vos utilisateurs envoient à votre bot. Pour plus d'informations, consultez la section Affichage des statistiques relatives aux énoncés.	22 septembre 2021
Nouvelle fonction	Amazon Lex V2 prend désormais en charge la langue coréenne (Corée). Pour plus d'informations, consultez Langues et paramètres régionaux pris en charge par Amazon Lex V2.	9 septembre 2021
Nouvelle fonction	Amazon Lex V2 fournit désormais un type de slot intégré pour les codes postaux britanniques. Pour plus d'informations, consultez AMAZON.UK. PostalCode	27 Juillet 2021
Nouvelle fonction	Amazon Lex V2 prend désormais en charge la langue anglaise (indienne). Pour plus d'informations, consultez Langues et paramètres régionaux pris en charge par Amazon Lex V2.	15 juillet 2021

Nouvelle fonction	Amazon Lex V2 fournit désormais un outil permettant de migrer un bot d'Amazon Lex V1 vers l'API Amazon Lex V2. Pour plus d'informations, consultez la section Migration d'un bot dans le manuel Amazon Lex Developer Guide.	13 juillet 2021
Nouvelle fonction	Amazon Lex V2 vous permet désormais d'accepter plusieurs valeurs pour un seul emplacement en anglais (États-Unis). Pour plus d'informations, consultez la section Utilisation de plusieurs valeurs dans un emplacement.	15 juin 2021
Nouvelle fonction	Vous pouvez désormais créer des robots plus grands pour les langues anglaises. Pour de plus amples informations, veuillez consulter Quotas .	11 juin 2021
Nouvelle fonction	Utilisez les politiques basées sur les ressources Amazon Lex V2 pour gérer l'accès à vos robots et à leurs alias. Pour plus d'informations, consultez la section Politiques basées sur les ressources dans Amazon Lex V2.	20 mai 2021

Nouvelle fonction	Amazon Lex V2 vous permet désormais d'importer et d'exporter des robots et des paramètres régionaux de robots. Vous pouvez utiliser cette fonctionnalité pour copier les robots et les paramètres régionaux des robots entre les comptes et AWS les régions. Pour plus d'informations, consultez la section Importation et exportation.	18 mai 2021
Expansion de région	Amazon Lex V2 est désormais disponible au Canada (centre) (ca-central-1).	17 mai 2021
Nouvelle fonction	Amazon Lex V2 prend désormais en charge les paramètres régionaux japonais (Japon). Pour plus d'informations, consultez Langues et paramètres régionaux pris en charge par Amazon Lex V2.	1 avril 2021
Nouvelle fonction	Amazon Lex V2 prend désormais en charge trois nouveaux types d'emplacements intégrés : `AMAZON.City` `AMAZON.Country` , et`AMAZON.State` .	12 mars 2021
Nouveau guide	Il s'agit de la première version du guide de l'utilisateur Amazon Lex V2.	21 janvier 2021

Référence d’API

La référence d'API est désormais un document distinct.

Glossaire AWS

Pour connaître la terminologie la plus récente d'AWS, consultez le Glossaire AWS dans la Référence Glossaire AWS.

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.